제2권 통권2호 1989년 1월 1일 발행(매월 1일 발행) 1988년10월10일 등록(등록번호 라-3794) ㊋140-150 서울특별시 용산구 갈월동 17

퍼스널컴퓨터
하이테크

89/01

특집/
IBM-PC ROM BIOS의 분석

기획/ **애플과 IBM-PC와의 통신**

제작코너/
- 31개 음의 표현이 가능한
  전2중 1200bps용 카드형 모뎀
- PC통신의 새로운 장을 여는
  음성출력시계



구입가이드

시스템 분석/ **시스템 자원**

애플 데스크탑 버스

테크닉/
- DOS의 기능 호출
- 디스크 드라이브의 응급처치
- 터보 파스칼의 강력한 기능들

활용/
- 에러처리루틴의 구성
- 하드디스크의 설치
- 메모리 확장

초보자들을 위한 페이지를 강화하고자 합니다.

최근들어 국내 퍼스널컴퓨터 시장에서도 놀라운 변화가 일어나고 있읍니다.

몇몇 기업에서 저가격의 제품을 내놓기 시작하더니 너도 나도 앞을 다투어 보급형 제품들을 선보였읍니다. 말그대로 누구나가 싼가격에 좋은 제품을 구입할 수 있는 그러한 시대가 온 것입니다. 그러나 이러한 보급형이란 말들이 말로만 끝나는 허울좋은 겉치레가 아니길 바랍니다.

예를 들어 모업체에서는 보급형 제품을 내놓으면서 그럴듯하게 선전을 하고 있읍니다. 그러나 알고보면 이 보급형이란 제품은 말뿐입니다. 실제로 컴퓨터를 사용할 수 있게 악세사리를 전부 갖추려면 선전하는 가격보다 훨씬 많은 금액이 소용됩니다. 이런 것이야말로 정말 유저를 우롱하는 처사가 아니라고 할 수 없읍니다.

이런 것들은 아직 컴퓨터마인드화가 이루어지지 않아서 생겨난 결과가 아닌가 생각됩니다. 제대로 유저들에게 인식되어 있다면 이런 편법을 사용하지 못할 것입니다.

예로써 저희 잡지는 지난 달에 창간호를 내보낸후 수많은 편지를 받았읍니다. 그런데 대부분이 한결같이 컴퓨터를 활용하고 싶은데 어떻게 해야될지 모르겠다는 내용들 이었읍니다. 컴퓨터는 접하면서도 아주 기초적인 내용조차 모르는 사람들도 많이 있었읍니다.

이번 달에도 지난 달과 마찬가지로 다양한 내용들은 게재하려고 노력하였지만 아직까지는 저희들의 욕심에 미치지 못하는 것 같읍니다.

이번 달에 들어간 내용중 가장 두드러진 기사는 IBM-PC ROM BIOS의 분석과 애플과 IBM-PC와의 통신이란 기사입니다.

최근들어 IBM-PC에 대한 관심이 더욱 더 높아져 가고 있읍니다. 이럴때 ROM BIOS에 대한 분석기사는 초보자에게는 다소 시스템을 이해하기가 어려운 느낌도 들지만 제대로 컴퓨터를 활용하고자 하는 사람들에게는 매우좋은 내용이라고 생각됩니다. 또 요즘 컴퓨터간의 통신이 매우 활발히 이루어지고 있읍니다. 아마도 머지않은 장래에 대부분의 사람들이 전화기와 마찬가지로 일상생활에 컴퓨터통신을 이용하지 않을까 생각됩니다.

그러나 아직까지는 대부분의 사람들이 컴퓨터를 가지고 다른 사람과 통신을 하고 싶으면서도 몰라서 접근못하는 경우가 대부분입니다.

이럴때 이 기사는 그러한 궁금증을 쉬원하게 해결해 주었다고 할 수 있읍니다.

다음달부터는 책의 내용이 많이 바뀌게 됩니다. 초보자들을 위한 쉽고 다양한 내용들이 많이 추가되며 전문인을 위한 페이지도 증설됩니다. 우선 다음달에는 초보자를 위하여 처음 컴퓨터를 배울때 어떻게 해야 되는지등 컴퓨터입문에 대한 내용을 도해식으로 쉽게 설명해 나가고자 합니다. 그리고 MS-DOS에 대한 내용도 간략하게 줄여 3회 정도의 분량으로 게재한 예정이며 그밖에도 현장취재기사들이 많이 강화됩니다.

독자개개인의 기호에 맞는 좋은 책이 되고자하오니 많은 관심이 있으시길 바라며 새해를 맞이하여 독자 여러분의 가정에 만복이 깃드시길 바랍니다.

화일의 초기화와 백업

컴퓨터 사용자가 어떤 좋은 응용 프로그램을 가지고 있다면 그 프로그램을 실행해보고 싶어질 것이다. 이러한 응용 프로그램의 실행은 시스템 자원을 관리하고 응용프로그램의 실행과 그 전개에 도움을 주어 컴퓨터와 사용자간의 인터페이스의 역할을 수행하고 있는 도스를 통하여 할 수 있다.

그러면 여기서 응용프로그램을 이용하기 위해 그 프로그램을 컴퓨터 메모리로 옮겨 주는 것을 도와주는 명령어와 화일이 어떻게 만들어지고, 특정 서브디렉토리에 있는 화일의 목록을 어떻게 끌어 낼 수 있으며, 화일의 백업 및 프로시듀어들을 자동적으로 처리하는 방법에 대해서 알아보자.

응용프로그램을 사용하려면

응용프로그램을 사용하려면 응용프로그램과 사용하고 있는 컴퓨터의 하드웨어적인 상태를 서로 일치시켜 주어야 하는데 이런 과정이 초기화(Install) 작업이다.

여기에서는 도스상태에서 매우 유용하게 프로그램을 초기화하고 실행시킬 수 있는 로터스 1-2-3*의 초기화 단계를 예로 들어 설명하겠다. 로터스 1-2-3의 초기화 단계는 다른 프로그램의 초기화 단계와 유사하기 때문에 이 예가 도움이 될 것이다.

여기서 사용할 로터스 1-2-3(릴리즈 2)은 각종 계산기능, 데이터처리기능, 그래프작성기능, 문서작성기능 등이 수록되어 있는 시스템 디스켓, 그래프를 프린터로 인쇄하는 기능이 수록된 PrintGraph 디스켓, 1-2-3을 사용하고 있는 컴퓨터의 하드웨어적 상태에 맞게 조정해 주는 프로그램과 1-2-3 화일과 다른 패키지의 화일을 서로 변환시켜주는 프로그램이 수록된 유틸리티 디스켓, 초기화를 위한 프로그램이 수록된 인스톨 라이브러리 디스켓, 1-2-3을 처음 사용하는 초보자를 위한 A View of 1-2-3 디스켓등 모두 5장의 디스켓으로 구성되어 있다.

개인용 컴퓨터 시스템은 고정되고 표준화된 장비들을 가지고 있지 않기 때문에 사용자의 요구에 맞게 하드웨어 소자들을 선택해야 한다.

그러므로 로터스 1-2-3와 하드웨어, 서로간의 호환성을 이루기 위한 초기화작업은 인스톨 라이브러리에 있고 사용하고 있는 모니터와 프린터와 같은 하드웨어와 로터스 1-2-3와의 중계를 담당하고 있는 프로그램인 여러 드라이버 프로그램중, 사용하고 있는 하드웨어 시스템에 적합한 드라이버 프로그램을 사용하여 이를 로터스 1-2-3 시스템 디스크와 PrintGraph 디스크, A View of 1-2-3 디스크에 복사하는 작업이다.

먼저 프로그램을 초기화하기 전에 다음사항을 체크해 보길 바란다.

1. 사용하고 있는 모니터가 칼라인가? 모노크롬인가?
2. 그래픽이 가능한가?
3. 프린터의 이름과 모델 번호는?
4. 사용하고 있는 도스의 버전은?
5. 두개의 플로피디스크 드라이브인가? 하드디스크 시스템인가?

초기화 작업을 하는 동안 수행될 작업들의 개요는 다음과 같다.

1. 도스화일들과 로터스 1-2-3 화일들을 전송한다.
2. 두개의 플로피시스템:도스화일들이 로터스 1-2-3 프로그램 디스켓으로 전송되어야 한다.
3. 하드디스크 시스템:로터스 1-2-3 프로그램 화일들이 하드디스크로 이동되어야 하드디스크에서 프로그램을 실행시킬 수 있다.
4. 사용하고 있는 시스템의 다양한 하드웨어 소자들과 호환성있게 로터스 1-2-3 프로그램을 만들어야 한다.

플로피디스크 시스템의 초기화

이제 두개의 플로피디스크 시스템에서 로터스 1-2-3 프로그램을 초기화하는 순서를 단계별로 알아보자.

우선 명령어를 치기전에 드라이브 A에는 도스 디스켓을, 드라이브 B에는 로터스 1-2-3 시스템 디스켓을 넣는다. 그리고 다음과 같이 타자하면

```
A>B:INSTALL
```

인스톨 화면이 나타난다(화면 1 참고).

* 로터스 1-2-3은 1982년에 발표된 계산기능, 그래프작성기능, 데이터처리기능, 문서작성기능을 가진 통합 소프트웨어이다.

〔화면 1〕

```
A>rem   *-------------------------------------------------------
A>rem  *      DOS INSTALLATION for 1-2-3 SYSTEM DISK     *
A>rem   *-------------------------------------------------------
A>rem  *      Make sure that:         *
A>rem   *       (1) A DOS 3.30 disk is in Drive A.
A>rem  *       (2) The 1-2-3 SYSTEM DISK is Drive  B.         *
A>rem  *              *
A>rem  *       If this is NOT so, then terminate this batch job now:
A>rem  * - Hold [Ctrl] and press "C".  Then press "Y".        *
A>rem  *    Otherwise . . .       *
A>rem   *-------------------------------------------------------
A>pause
Strike a key when ready . . .
```

아무키나 누르면 초기화 프로그램은 자동적으로 다음 명령을 화면에 표시한다.

```
A>sys b:
System transferred
```

여기에서 "System"의 약자 "sys"명령어는 지정한 디스켓에 시스템화일을 옮기는 역할을 한다. 이 화일들은 키보드의 입력을 해석하는 일이나, 모니터에 화면을 나타내거나 디스크드라이브들 간의 원활한 상호작용을 돕는 것과 같은 중요한 작업들을 수행 가능케 하는 명령어이다. 그러므로 이 시스템화일은 대단히 중요하고, 사용상의 부주의로 화일이 깨질 수 있기 때문에 사용자에게 히든화일로 은폐되어 있고 접근이 불가능하다.

다음 명령어는 로터스 1-2-3 시스템 디스크에다 명령어처리 화일을 복사한다. 이 화일은 시스템을 시작시키고 시간과 날짜의 프롬프트를 나타내 준다.

```
A>copy COMMAND.COM B:   1 File(s) copied
```

이 명령은 COMMAND.COM이란 화일을 B 드라이브에 같은 이름으로 복사하는 것을 의미한다.

다음 단계는 모니터를 초기화시키는 것인데 로터스 1-2-3 프로그램은 모노크롬화면, 칼라화면, 그래픽능력을 갖춘 모노크롬(허큘리스), 모노크롬과 칼라모니터 등을 인식할 수 있다.

드라이브 B에는 로터스 1-2-3 시스템 디스크를, A에는 로터스 1-2-3 유틸리티 디스크를 넣는다. 드라이버들은 로터스 1-2-3 프로그램 디스켓의 모든 내용을 초기화할 것이며 처리하는 과정에서 디스크를 변경시킨다. 예를 들어 칼라 모니터를 가지고 있다면 다음 화면 2와 같이 리스트한다.

그 이외의 적당한 드라이버 화일들은 로터스 1-2-3 유틸리티 디스크에서 선택된다. 다음 단계는 로터스 1-2-3 시스템 디스켓에 이런 화일들을 복사하는 것이며 그 처리과정은 자동적으로 계속된다. (화면 3)

하드디스크 시스템의 초기화

이제 하드디스크 시스템에 대한 초기화 작업을 해보자. 하드디스크 시스템은 플로피디스크 시스템과는 저장용량이 다르기 때문에 초기화 프로시듀어를 시작하기 전에 하드디스크의 주위환경(Enviornment)을 살펴보자.

보통의 플로피 디스켓은 360KB, 고밀도 플로피 디스켓은 1.2MB정도 데이터를 수용할 수 있다. 그러나 하드디

〔화면 2〕

```
A>COLOR B:
A>rem   *-------------------------------------------------------
A>rem        "COLOR"  DRIVE INSTALLATION
A>rem   *-------------------------------------------------------
A>rem  *       Be sure that you have read GETTING STARTED.   *
A>rem  * Make sure that the source drive is the "default  drive"
A>rem  *            and that "B:" is the target drive.
A>rem  *
A>rem  *    If this is NOT so, then terminate this batch job now:
A>rem  *             - Hold [Ctrl] and press "C"       *
A>rem  *             - Press "Y"
A>rem  *    Otherwise . . .
A>pause *-------------------------------------------------------
Strike a key when ready ...
A>copy   IBM0COLO.DRV   TD.DRV
     1 File(s) copied
A>copy   IBM1G2.DRV     GD.DRV
     1 File(s) copied
A>copy   IBM2KB.DRV     KB.DRV
     1 File(s) copied
A>copy   IBM3PR.DRV     PR.DRV
     1 File(s) copied
```

〔화면 3〕

```
A>continue  COLOR B:
A>rem   *---------------------------------------------------
A>rem  *  TWO-DISKETTE INSTALLATION  *  HARD-DISK INSTALLATION
A>rem   *---------------------------------------------------
A>rem  *                             *                      *
A>rem  *        Place your           *  The next step will  *
A>rem  *     1-2-3 SYSTEM DISK       *  copy the drivers to *
A>rem  *       in drive "B".         *  drive "B:".          *
A>rem  *                             *  Then, you will be   *
A>rem  *                             *  instructed to terminate
A>rem  *                             *  this batch job.     *
A>rem  *                             *                      *
A>pause  *---------------------------------------------------
Strike a key when ready ...
A>copy   ??.DRV  B:
TD.DRV
GD.DRV
KB.DRV
PR.DRV
     4 File(s) copied
          ...
```

스크는 수십MB 이상을 저장할 수 있기 때문에 수백에서 수천개 이상의 화일들을 저장할 수 있다. 그러나 이렇게 많은 화일들을 저장하고나서 특정 화일들을 찾을때는 매우 복잡하다는 문제점을 안고 있다.

이러한 문제점을 해결하기 위해 도스에서는 화일들을 그룹별로 분류한 디렉토리에 기억시켜 특정화일을 찾는데 소요되는 시간을 줄일 수 있게 했다. 디렉토리는 각각의 서브디렉토리로 구성되어 있고 이들의 구조는 **그림 1**과 같이 루트디렉토리에서 서브디렉토리가 가지치기 형식으로 연결되는 트리구조로 되어 있다.

루트디렉토리는 디렉토리의 가장 높은 레벨이고 사용자가 디스크를 포맷할 때 도스는 그 디스크상에 루트디렉토리를 만든다.

□ 주의

만일 하드디스크 시스템을 사용하고 있다면 서브디렉토리를 만들고 그 프로그램에 대한 데이터 화일에 대해 두번째의 디렉토리를 만든다.

서브디렉토리는 도스 명령어 MKDIR를 이용하여 만들 수 있다. 로터스 1-2-3 프로그램에 대한 서브디렉토리를 만들려면

```
MKDIR\LOTUS
```

라고 하는데 이때"\"표시는 루트디렉토리를 말한다.

하드디스크를 사용하고 있다면 그것을 사용하기 전에 각 응용 프로그램을 하드디스크로 복사해야 한다.

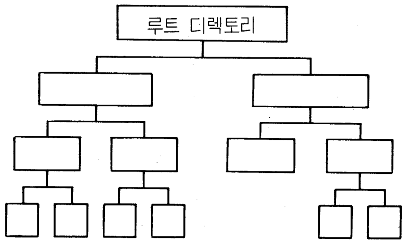
초기화 프로시듀어의 처음 단계는 하드디스크에 있는 서브디렉토리로 모든 로터스 1-2-3 화일을 복사하는 것인데 여기에는 로터스 시스템, PrintGraph, 유틸리티 등 세가지 프로그램 디스켓이 포함된다.

이제 드라이브 A에 로터스 1-2-3 시스템 디스켓을 넣고 다음 **화면 4**와 같이 한다.

이와같은 형태로 계속해서 다른 시스템디스켓도 카피한다.

로터스 1-2-3 프로그램이 실행되는 동안 새 디스크를 포맷팅하는 것과

〔그림 1〕 디렉토리와 서브디렉토리



〔화면 4〕

```
C>COPY A:*.*
A:123.EXE
A:123.HLP
A:123.CNF
A:LOTUS.COM
A:LTSLOAD.COM
A:AUTOEXEC.BAT
A:INSTALL.BAT
A:TD.DRV
A:GD.DRV
A:KB.DRV
A:PR.DRV
     11 File(s) copied
```

한쪽 디스크에서 다른 디스크로 복사하는 등 도스의 기능을 사용할 수 있다. 로터스 1-2-3 프로그램에서 이러한 명령어들중의 하나를 사용하려면 적당한 도스화일에 접근할 필요가 있으며, 이럴때는 로터스 서브디렉토리에서 화일을 찾아야 한다.

도스디스켓을 드라이브 A에 넣고 필요한 도스 화일을 로터스 서브디렉토리로 복사할 때 다음 **화면 5**와 같이 한다.

다음 마지막 단계는 로터스 1-2-3 드라이버들을 초기화하는 것이다. 마지막 명령어를 입력하기 전에 지정 드라이브를 **화면 6**과 같이 C에서 A로 바꿔 주어야 한다.

초기화 절차가 끝났으면 화면의 지시에 따른다.

□ 주의

프로그램을 복사하기전에 원본 프로그램에 있는 쓰기-방지(write-protect) 실버테이프를

〔화면 5〕

```
C>COPY A:COMMAND.COM
     1 File(s) copied

C>COPY A:FORMAT.COM
     1 File(s) copied

C>COPY A:CHKDSK.COM
     1 File(s) copied

C>COPY A:DISKCOPY.COM
     1 File(s) copied

C>COPY A:DISKCOMP.COM
     1 File(s) copied
```

〔화면 6〕

```
C>A:
A>COLOR C:
A>rem   *---------------------------------------------------
A>rem      "COLOR" DRIVER INSTALLATION
A>rem   *---------------------------------------------------
A>rem   *      Be sure that you have read GETTING STARTED.
A>rem   * Make sure that the source drive is the "default  drive"
A>rem   *      and that "C:" is the target drive.
A>rem   *
A>rem   *   If this is NOT so, then terminate this batch job now:
A>rem   *   - Hold [Ctrl] and jpress "C"
A>rem   *   - Press "Y"
A>rem   *   Otherwise...
A>pause *---------------------------------------------------
Strike a key when ready...
```

붙이지 않으면 부주의할 경우 데이터가 지워지는 실수를 범할 수 있다.

대부분의 디스켓은 한면의 끝에 작은 직사각형 모양의 비어낸 부분을 가지고 있다. 이 비어낸 부분을 싸지 않으면 디스켓에 내용을 쓸 수 있게 된다. 이 부분에 실버테이프로 싸주면 쓰기방지가 된다. 이렇게 보호된 디스켓에 기록을 시도하게 되면 다음과 같은 에러 메시지를 받게 된다.

```
Write Protect Error Writing Drive
Abort, Retry, Ignore?
```

일단 복사가 완성되면 쓰기-방지 실버테이프를 붙이도록 하라. 이러한 보호법은 실수에 의한 포맷팅이나, 영구적으로 프로그램 디스켓을 보호하고 싶을 때 주요하게 쓰인다.

□ 주의

플로피디스크 시스템에서는 소프트웨어 프로그램을 사용할때 항상 사용 가능하게 디스켓을 포맷해 놓아야 한다.

스프레드 시트같은 응용 프로그램으로 화일을 만들려고 할때 FORMAT과 같은 도스명령어들은 프로그램을 빠져나와야 실행시킬 수 있기 때문에 직접 디스켓을 저장해 놓지도 못하고 중간에 포기해야 되는 일이 발생한다. 그러므로 디스켓은 구입할 때마다 데이터를 저장할 수 있게 모두 포맷팅 작업을 해 놓아야 한다.

□ 주의

플로피디스크 시스템에서 도스 디스켓은 보통 FORMAT 명령어를 칠때 지정드라이브에 그 화일이 있어야 한다.

도스명령어를 수행하여 프로그램 화일에 접근할 수 있는데 어떤 도스 명령어는 도스 명령처리 프로그램내부에 있는 서브프로그램들이고 다른 명령어들은 도스디스켓에 별개의 화일들로 존재한다.

FORMAT 명령어는 외부화일이다. 즉 이것은 도스 디스켓이 지정 드라이브에 있음을 의미하고, 따라서 도스는 포맷하는 프로시듀어를 수행하기 위해 필요한 정보를 끌어내야 한다. 만일 FORMAT 프로그램화일을 찾을 수 없다면 도스는 다음과 같은 메시지를 나타낸다.

```
Bad command or filename
```

이것은 도스가 명령어 화일을 찾지 못함을 말하며 도스는 항상 입력된 이름으로 화일을 찾기 때문에 명령어의 이름을 잘못 쳤을 때도 이런 메시지가 나타난다.

프로그램의 수행

□ 주의

프로그램을 시작하려면 도스 프롬프트 상태에서 프로그램의 이름을 간단히 친다. 로터스 1-2-3을 수행하려면 도스 프롬프트 상태에서

```
A>LOTUS
혹은
C>LOTUS
```

라고 하면 곧 디스크 드라이브에 빨간 불이 들어오며 회전하는 소리가 들리게 됨과 동시에 로터스 메인 메뉴가 나타난다.

□ 주의

만일 프로그램을 친후"Bad command or filename"이란 메시지가 나타나면 프로그램 이름이 잘못 되거나 현재 지정 드라이브/디렉토리 조합이 잘못 지정되었을 경우이다.

하드디스크에 로터스 1-2-3 프로그램을 가지고 있으나 현재 지정 드라이브가 A라고 가정하자. 만일 로터스 1-2-3를 메모리로 옮기기 위해 로터스 명령을 치면"Bad command or filename" 이란 메시지가 나타난다. 왜냐하면 지금 로터스 1-2-3 프로그램은

C드라이브에 있고 현재 지정 드라이브는 A이기 때문이다. 따라서 C드라이브로 드라이브 변경을 한 후 명령어를 친다.

만일 하드디스크 시스템이나 도스 버전 3.3 플로피디스크 시스템을 가지고 있다면 그 디스크에 하나 이상의 서브디렉토리를 만들 수 있다. 이런 경우 현재 지정 드라이브가 정확해야 될 뿐만 아니라 현재 서브디렉토리도 정확해야 한다.

서브디렉토리를 변경하는데는 변경 디렉토리 명령어 CD를 사용하여 다음과 같이 한다.

```
CD C:\LOTUS
```

□ 주의

하드디스크 시스템에서 만일 프로그램 이름을 친 후 도스 프롬프트나 이상한 메시지가 나타나면 이런 경우는 프로그램이 보호된 것이거나 제대로 카피되지 않은 것이며 이러한 것은 그대로 사용할 수 없다.

일반적으로 이런 경우는 프로그램을 단지 하드디스크(다른 플로피 디스켓이 아닌)로 복사할 수 있음을 의미하고 원본 프로그램 디스켓을 하드디스크로 옮길 때는 사용환경에 따라 다르게 인스톨시켜줘야 한다.

디렉토리내의 화일을 보려면

특정 디렉토리내의 화일을 보고싶을 때는 DIR 명령어를 사용한다. 이는 특별히 지정된 디스크 서브디렉토리에 저장된 화일의 목록을 나타내준다. 또한 크기를 포함하여 화일에 관한 정보를 얻을 수 있고 시간, 날짜별로 화일이 만들어진 것이나 수정된 때를 알 수 있다.

서브디렉토리에 있는 화일의 목록을 얻기 위해서는 DIR이라는 명령어를 사용한다.

□ 주의

화면에 한번에 나타내기에는 너무 긴 화일들의 목록을 보고 싶을때 DIR 명령에 PAUSE나 WIDE 란 모드를 덧붙여 사용한다.

/P는 한 화면에 화일의 목록이 다 차지하게 되면 저절로 멈추고 스크린 맨하단에 아무키나 누르면 계속된다는 메시지가 나온다.

/W는 단지 화일의 이름만으로 디렉토리와 함께 나타내주기 때문에 화면 7과 같이 한화면에 다 표시하게 된다. 그러므로 각 화일 요소들의 크기나 날짜, 시간 등은 생략하게 된다.

□ 주의

서브디렉토리에 있는 화일의 일부분만을 보고 싶을때에는 DIR 명령어와 와일드카드(wild card)를 사용한다.

현재 서브디렉토리에서 화일 확장자명이".EXE"로 된 화일을 가지고 있는 모든 화일들을 찾고 싶다면"*"라는 와일드 카드문자를 사용한다. 다음화면 8의 예를 보자.

즉 EXE라는 확장자를 가지고 있는 모든 화일의 목록이 표시되게 된다. 또 만일 두번째 자리에"O"라는 문자를 가진 4문자 화일이름의 모든 목록을 보고 싶으면"?"라는 와일드 카드 문자를 사용한다."?"의 의미는 이 위치에 있는 어느 문자든지 받아들이라는 뜻이다. 화면 9의 예를 보자.

〔화면 7〕

```
A>DIR/W

 Volume in drive A is MS-DOS
 Directory of  A:\

COMMAND  COM    JOIN     EXE    KEYB     COM    KEYBOARD SYS    LABEL    COM
LCD      CPI    MODE     COM    MORE     COM    MORTGAGE BAS    NLSFUNC  EXE
PRINT    COM    PRINTER  SYS    RECOVER  COM    REPLACE  EXE    RESTORE  COM
SD       EXE    SELECT   COM    SHARE    EXE    SORT     EXE    SUBST    EXE
SYS      COM    TREE     COM    VDISK    SYS    XCOPY    EXE    BASIC    COM
BASIC    PIF    BASICA   COM    BASICA   PIF
        28 File(s)     32768 bytes free
```

〔화면 8〕

```
A>DIR *.EXE

 Volume in drive A is MS-DOS
 Directory of  A:\

JOIN     EXE     8969   3-17-87  12:00p
NLSFUNC  EXE     3060   3-17-87  12:00p
REPLACE  EXE    11775   3-17-87  12:00p
SD       EXE     1280  11-11-83  10:10a
SHARE    EXE     8608   3-17-87  12:00p
SORT     EXE     1977   3-17-87  12:00p
SUBST    EXE     9909   3-17-87  12:00p
XCOPY    EXE    11247   3-17-87  12:00p
         8 File(s)     32768 bytes free
```

〔화면 9〕

```
A>DIR ?O??.*

Volume in drive A is MS-DOS
Directory of  A:\

JOIN     EXE      8969   3-17-87  12:00p
MODE     COM     15487   3-17-87  12:00p
MORE     COM       313   3-17-87  12:00p
SORT     EXE      1977   3-17-87  12:00p
        4 File(s)     32768 bytes free
```

화일의 백업과 저장

작업하는 동안 데이터 화일을 저장하게 되면 후에 다시 첨가하고, 수정할 수 있다. 화일을 저장하는 것은 단지 몇초 동안에 일어나는 작업이다. 그러나 잃어버린 화일을 다시 만드는데는 얼마나 많은 시간을 소요해야 되는지 생각해보라.

다음의 예는 컴퓨터를 사용하다가 종종 접하는 문제들로 응용 프로그램으로 작성한 데이터를 일부 혹은 전부 잃어버릴 수 있다.

1. 전원상실

몇초 동안의 전원상실은 사용자가 막 끝마친 작업내용을 잃어버리게 된다. 이것은 단순히 금방 만들었던 화일 하나를 잃어버린다는 의미일 수 있다. 그러나 만일 정보를 저장하는 과정에서 전원이 상실되면 전체 디스크에 중요한 상실을 가져온다. 이것은 현재 화일이 파괴되는 것 이상을 의미한다.

2. 하드웨어 상실

컴퓨터 하드웨어는 보통 매우 신빙성이 있다. 그러나 소자들 중의 하나가 완전히 제기능을 발휘하지 못할 수 있는데 특히 디스크드라이브가 가장 고장나기 쉬운 부분이다. 왜냐하면 그것에는 움직이는 부분이 있기 때문이다.

디스크드라이브에 관한 문제들은 데이터를 파괴하는 결과를 초래할 수 있다.

3. 움직이지 않는 프로그램

화면이 멈추거나 디스크 드라이브 헤드가 움직이지 않거나 계속 돌기만 하는 등 프로그램 중간에 컴퓨터가 멈추면 어느 키를 쳐도 아무런 효과가 없다. 이런 종류의 문제는 보통 프로그램에 결함(bug)이 있기 때문이다. 이렇게 되면 작업과정동안 만들었던 모든 작업을 잃게 된다.

4. 배드(Bad) 디스크

지나치게 심한 디스크 조정작업은 디스크에 물리적인 손상을 입히게 되어 전체 디스크의 내용을 읽을 수 없게 된다.

5. 잘못된 전원코드

전원코드가 보행자들이 다니는 길에 있으면 부주의로 전원에 이상을 줄 수 있다.

이러한 문제들로 인해 시스템에게 100% 모든 것을 의존할 수는 없다. 그러므로 기능이 좋은 백업프로시듀어를 이용하여 화일을 백업하게 되면 이런 불행한 결과를 줄일 수 있다.

백업 화일을 만드는 것은 플로피디스크들과 하드디스크들 사이에 있는 화일의 복사본을 만드는 간단한 처리이다.

□ 주의

사용자가 작업하는 응용 프로그램으로 직접 현재 화일의 백업 화일을 만들 수 있다.

예를 들어 로터스 1－2－3에서는 사용자의 작업을 /File Save 명령으로 저장할 수 있다. 언제든지 화일을 만들어 그것을 하나 이상의 디스켓에 저장할 수 있다. 이 과정은 키 하나로 즉시 화일을 저장할 수 있는 빠르고 쉬운 방법이다.

□ 주의

만일 중요한 데이터 화일의 백업을 두 세개로 가지고 있지 않았다면 때때로 데이터를 잃어버릴 수 있다.

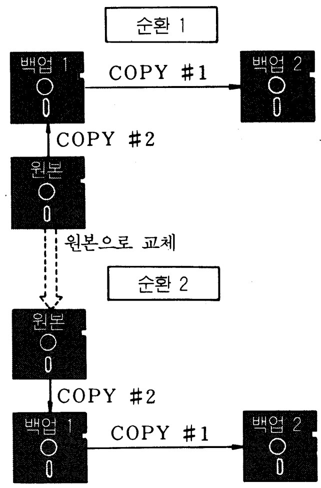
즉, 이 말은 만일 한 화일이 어떤 방법으로 못쓰게 되었다면 그것을 백업하는 과정은 그 화일의 최종 복사본위에 쓰는 (write) 것이다. 이런 문제점을 해결하기 위해 그림 2처럼 두개의 백업 디스켓을 갖고 있어야 한다.

□ 주의

하드디스크 시스템에서는 한 플로피 디스켓에서 다른 곳으로 복사하기 위한 전송 구역이 되는 일시적인 서브디렉토리를 만들 수 있다.

만일 지금 하드디스크 시스템을 사용하고 있다면 대부분 하나의 플로피 디스크를 가지고 있기 때문에 하드디스크에 일시적으로 전송구역이 될 수 있게 서브디렉토리를 만든다. 그 다음에 COPY 명령으로 하드디스크에 옮긴다.

〔그림 2〕 일반적인 백업 절차



그 다음에 COPY 명령으로 하드디스크에 있는 내용을 다른 플로피 디스켓에 옮긴다. 일이 다 끝났으면 서브디렉토리에 있는 모든 화일을 지운다. 다음 **화면** 10의 예를 보자.

사용자는 이런 처리과정을 하나의 배치화일로 명령어들을 나열하여 더욱더 간단히 처리할 수 있는데 이것은 나중에 좀더 자세히 알아보자.

□ 주의

COPY 명령어를 사용하여 하드디스크의 내용을 플로피디스크에 백업(BACK UP)할 수 있다.

하드디스크 시스템을 사용하고 있을 때 만일 하드디스크가 깨진다면 그곳의 화일들은 영영 복구할 수 없게 된다. 이런 점을 미리 생각하여 하드디스크에 있는 중요한 데이터는 플로피디스크에 옮겨 넣는다. 적은 수의 화일들을 백업하기 위한 가장 간단한 방법은 하드디스크에서 서브디렉토리로 들어가 그곳의 내용을 플로피디스크에 옮기는 것이다.

예를 들면 **화면** 11의 예와 같다.

이것은 하드디스크에 있는 BUDGDATA 서브디렉토리에 있는 모든 화일들을 A드라이브에 있는 플로피디스크에 복사한다. 그러나 이 서브디렉토리에 있는 화일들은 길이가 너무 길어 플로피디스크의 구역을 넘을 수도 있다. 만일 모든 화일이 복사되기도 전에 이런 일이 발생하면 하나 이상의 디스켓을 사용한다거나 화일들을 디스크에 기록하는 등의 성가신 작업을 해야 한다.

도스에서는 하드디스크 화일들을 플로피디스크에 백업하는 과정을 손쉽게 할 수 있는 특별한 BACK UP과 RESTORE 명령을 가지고 있다. 그러나 이러한 명령어들은 하드디스크 전체의 내용을 플로피에 백업할 때 매우 번거롭다. 이때 유동 테이프 카트리지나 유동 윈체스터 디스크 카트리지로 백업하는 것이 대체로 편리하며 실제로 큰 해결책이 되고 있어 하드디스크의 백업용도로 높은 가치를 지니고 있으며 추가적인 온라인 기억장치를 사용하게 하는 장점을 가지고 있다.

도스 프로시듀어의 자동수행

도스명령어들과 친숙해 감에 따라 많은 작업들이 대체로 도스명령어를 하나 이상 포함한 루틴프로시듀어로 되어 있음을 알 수 있다.

사용자는 배치화일이라는 곳에 이런 일련의 명령들을 조합해 놓으면 시간과 노력을 절약할 수 있고 배치화일의 이름만을 간단히 입력함으로써 작업을 실행시킬 수 있다. 배치화일들은 도스의 가장 유용하고 강력한 특징중의 하나이며 이것을 이용하면 유용한 데이터를 잃어버리는 것같은 큰 실수를 미리 방지할 수 있다.

□ 주의

만일 한 프로그램이 많은 시간을 사용한다면 부트될 때 자동적으로 프로그램이 메모리로 옮겨지는 AUTOEXEC.BAT 화일을 이용한다.

시스템을 처음 시작할 때 자동적으로 로터스 1-2-3 메모리로 옮겨지는 배치화일을 만들어 보자. 여기서 AUTOEXEC. BAT라는 배치화일을 이용하면 도스 명령어들의 목록들이 자동처리되어 일이 간단하게 처리된다. 우

화면 10

```
---------

C>MKDIR\TEMP

C>CD\TEMP

C>COPY A:*.*
A:BUDGET83.WKS
A:BUDGET84.WKS
A:BUDGET85.WKS
A:YTDACCTS.WKS
     4 File(s) copied

C>COPY *.* B:

Insert diskette for drive B: and strike
any key when ready

 BUDGET83.WKS
 BUDGET84.WKS
 BUDGET85.WKS
 YTDACCTS.WKS
    4 File(s) copied

C>ERASE *.*
```

화면 11

```
C>CD\BUDGDATA

C>COPY *.* A:
BUDGET83.WKS
BUDGET84.WKS
BUDGET85.WKS
YTDACCTS.WKS
   4 File(s) copied
```

선 화일이름을 AUTOEXEC. BAT라고 짓고 현재 지정디렉토리에 그 화일을 복사한 후 로터스 1-2-3 프로그램을 메모리로 옮긴다. 실제 프로시듀어를 만들어 보자.

```
COPY CON: AUTOEXEC.BAT
LOTUS
<F6> <ENTER>
```

여기서 마지막 줄은 F6 함수키를 치고 ENTER키를 치라는 뜻이며 도스에게 드라이브 A에 있는 디스켓에 배치화일을 저장하라고 지시하는 것이다. 이 프로시듀어를 시작하기 전에 A드라이브에는 로터스 1-2-3 시스템 디스켓이 있어야 한다. A드라이브에 있스켓이 있어야 한다. 배치화일을 시험해 보기 위해 로터스 1-2-3 시스템 디스켓이 A드라이브에 있는지 확인하고 CTRL,ALT,DEL 등의 3개의 키를 동시에 눌러 시스템을 재부팅시킨다.

도스가 부팅처리를 시작할 때 먼저 AUTOEXEC.BAT 화일이 있는가를 알아보고 존재하면 화일이 기입된 명령들을 읽음으로 해서 프로그램을 사용할 수 있게 된다.

〈용어해설〉

●시스템 자원(System Resource)
데이터 처리를 하는데 필요한 하드웨어, 정보, 인적 구성요소를 의미하며 하드웨어 자원은 CPU, 주기억장치, 채널, 입출력제어장치 등을 말하고 정보자원은 제어프로그램, 처리프로그램, 데이터 인적자원은 프로그래머, 오퍼레이터 등을 말한다.

●인터페이스(Interface)
2개 이상의 구성요소의 경계에서 그들에게 공용되는 부분을 말한다. 인터페이스는 범용과 전용으로 나눌 수 있는데 범용은 어떤 입출력기기에도 적합하도록 된 것이고 전용은 특정의 입출력기기에 적합하도록 된 것이다.

●화일(File)
정보처리의 과정에서 다루는 하나의 단위로서 취급되는 관련된 레코드(Record)의 집합을 가리킨다.

●패키지(Package)
어떤 용도를 위해 작성된 범용 프로그램을 지정한다. 예를 들면 재고관리 패키지, 급여계산 패키지, 각종 데이터베이스 패키지 등.

●라이브러리(Library)
특정의 문제나 그 일부를 해결하기 위해 표준화되고 테스트된 서브루틴 등을 모아서 체계화 한 것으로 라이브러리에 등록되는 서브루틴은 연계적인 편집이 쉽게 이루어지도록 상대형식 또는 소스(원시) 프로그램으로 되어 있다.

●드라이버(Driver)
주변장치나 그 인터페이스를 CPU에서 제어하는 작은 프로그램.

●프롬프트(Prompt)
OS가 오퍼레이터나 사용자에게 주는 메시지로서 키로부터의 입력이 가능하다는 것을 알린다.

●환경(Environment)
하나의 시스템이 주어졌을 때 그 시스템의 외부에 있는 모든 요소의 집합을 말한다.

●버그(Bug)
오류 또는 오동작을 말하며 프로그램의 에러를 말한다.

●백업 카피(Back-up Copy)
화일이나 데이터 세트의 복사를 말하며, 원래의 화일이나 데이터 세트가 파괴될 경우에 대비하기 위해 준비한다.

●기억장치(Memory Unit)
수치나 문자, 기호 등을 기억하는 장치로서 프로그램이나 데이터 또는 처리결과를 기억한다. 기억방법의 구성은 전기의 ON과 OFF에 대응한 1과 0의 조합으로 여러가지 값을 기억시킨다.

●윈체스터 디스크(Winchester Disk)
대용량의 고성능 고정 헤드 디스크.

●온라인(On-line)
중앙처리장치의 직접제어하에 있는 상태. 즉, 계산기 시스템에 있어서 주변장치나 외부장치가 CPU의 직접제어하에 있는 상태를 말한다. 특히 통신회선 등을 사용하여 단말을 접속하고 데이터 처리를 행하는 온라인 상태의 시스템을 온라인 시스템이라고 한다.

●루틴(Routine)
어떤 통합된 기능이 있는 프로그램으로 정의된다. 컴퓨터가 특정한 기능을 수행할 수 있도록 마련된 일련의 명령군으로 기능적으로 관련된 2개 또는 그 이상의 명령으로 된 프로그램의 단편을 의미한다. 보통 수행하여야 할 기능의 명칭앞에 붙여서 무슨 루틴이라고 호칭한다.

●워드(Word)
비트 몇개를 통합하여 하나의 단위로 한 것으로 보통 1워드는 16비트로 하지만 컴퓨터의 종류에 따라서 4, 8, 16, 32비트를 각각 1워드로 사용할 수도 있다.

●기억 용량(Storage Capacity)
기억장치에 기억할 수 있는 데이터의 양을 말한다. 일반적으로 비트, 바이트, 워드의 수로 용량의 크기를 표시하지만 바이트(1문자를 기억하는 크기) 수로 표시하는 경우가 많다.

●어드레스(Address)
기억장치에 데이터를 리드 혹은 라이트 할 때 기억장소의 위치를 지정해 주는 것.

●플로피디스크 장치(Floppy Disk Unit)
폴리에스텔 판 위에 자성재료를 칠하여 염화비닐 재킷에 넣은 기록매체를 플로피디스크라고 한다. 이 플로피디스크를 일정한 속도 회전시켜 자기 헤드로 데이터를 기록하거나 판독하는 장치를 말한다. 플로피디스크는 값이 싸고 데이터가 어느 곳에 있더라도 거의 같은 시간에 판독과 기록을 할 수 있고 다루기도 용이하기 때문에 널리 이용되고 있다.

●오버레이 구조(Overlay Structure)
주기억 장치에 한꺼번에 저장할 수 없는 큰 프로그램을 몇개의 부분으로 나누어 놓고 필요에 따라 오버레이(주기억에 연속적으로 유지되지 않는 구분) 형식으로 주기억 영역에 로딩되게 준비를 한 구조. 오버레이는 연계편집(linkage edit) 프로그램의 한가지 기능이며 오버레이 구조는 동시에 실행될 수 없는 여러개의 세그먼트에 컴퓨터 프로그램의 부분을 로드하는 기구이다.

●피드백(Feedback)
어떤 데이터를 입력, 처리, 출력하는 시스템에서 출력 정보가 목적에 일치할 때까지 정보를 입력쪽에 되돌리는 것을 말한다.

●프로시듀어(Procedure)
문제 해결을 위하여 취해지는 활동의 순서 또는 절차이며 프로그램중의 주요한 구문 단위의 하나이다.

●PC 레이다●

국내소식

-한국과학기술원
인공지능컴퓨터 국내 최초 개발

한국과학기술원 컴퓨터구조연구실 조정완박사팀은 사람과 같이 주어진 지식과 기억돼 있는 데이터를 사용해 스스로 추론할 수 있는 인공지능 컴퓨터를 국내최초 개발, 최근 시제품을 제작하는데 성공하였다.

인공지능 구현방식으로 널리 채택되고 있는 WAM(워렌추론머신)을 진일보시켜 병렬처리와 혼성지식표현이 가능하도록 설계된 동 시스템은 X-WAM이라고 명명하였으며 병렬처리를 실현함으로써 기존 인공지능보다 훨씬 빠른 속도로 추론할 수 있을 뿐만 아니라 혼성지식표현 시스템을 구현함으로써 복합연산, 복잡한 문제해결이 가능하다.

조박사팀은 지난 84년 인공지능 개발에 착수, 과기처 특정 연구과제로「차세대 컴퓨터시스템 개발연구」의 4차연도에서 X-WAM을 완성하였다.

동시스템은 CPU로 모토롤라사의 68000 마이크로프로세서를 채택하고 입출력제어 컴퓨터로 AT를 사용했다.

또한 동 개발팀은 5차년도 연구가 종결되는 89년 중반까지 AND 병렬처리외에 OR 병렬처리까지 구현할 수 있는 X-WAM2를 개발할 계획이다.

-대한전자공학회
PC 통신 관련 기술세미나 개최



지난 11월 29, 30일 양일간에 걸쳐 대한전자공학회 주최로 PC통신 관련 기술세미나 및 통신 시험발표회가 한국 종합전시장에서 개최되었다.

PC통신시스템이란 PC를 이용하여 전자메일, BBS 등의 서비스를 제공하고 기존 단말기와의 통신이 가능하도록 함은 물론이고 각종의 DB접속을 지원하는 시스템을 가리키는 것으로 현재 국내에서는 이기종 PC간의 통신 프로토콜의 표준화가 이루어지지 않은 상태이다.

이에 한국전자통신연구소(ETRI)에서는 PC통신방식 표준화연구를 수행중에 있으며 3개의 업체에서는 표준화된 통신부가장치(Adaptor)를 개발하였다.

한편 동 세미나에서는 PC통신기술의 세계적인 추세를 소개하고, 특히 KTA 의 표준규격이 될 가능성이 큰 PC통신규격(안)-ETRI 규격(안)-의 구체적인 설명이 있었다.

또한 현대전자, 금성통신, 삼성전자가 제작한 PC통신 어댑터끼리의 통신시험과 개발사례 설명회 등으로 진행되었다.

-한국전기통신공사
TDX-1A 비음성서비스기능 개발

통신공사는 현재 공급중인 국산디지탈 시분할교환기 TDX-1A에 고품질 및 56Kbps 급의 고속전송속도를 지닌 비음성서비스 기능의 개발을 완료, 시범서비스를 위한 규격제정과 기존에 구축된 행정통신망을 이용한 원활한 상용서비스의 구현에 박차를 가하고 있다.

국내 처음으로 개발된 이 TDX-1A의 비음성서비스 기능은 기존 전화망보다 20배나 빠른 전송속도와 고속의 문서전달을 비롯 선명도가 뛰어난 화상전화, 고속 텔리텍스, 비디오텍스 등 고속품질의 데이터 송신 등으로 꼽을

수 있다.

한편 동 서비스시스템은 지난 87년, 행정통신망 개발업체인 대우통신이 전자통신연구소로부터 비음성서비스용 단말기 개발업체로 선정받음으로써 DSLB (디지탈가입자 정합회로)를 비롯한 클럭공급, 측정장비 정합회로 팩, 관련 소프트웨어 등을 개발하면서 작년 하반기에 통신공사내의 사업개발단 기술실과 서대전전화국, 한국전자통신연구소등 4곳에 고속 팩시밀리 단말기를 연결함으로써 상용시험을 완료한 것으로 알려졌다.

또한 통신공사의「TDX에 의한 ISDN 시범사업계획」따르면 TDX 기종에 ISDN 기본기능개발을 착수, 내년까지 종합적인 시범망을 구축키로 하고 다가오는 5월에 개최되는 아시아 ISDN 협의회에 ISDN 기본기능을 지닌 TDX 교환기를 개발, ISDN 시범망을 구축하고 이를 토대로 교환시설 기능시험, 접속장치시험, 단말기 기능시험을 거쳐 본격 시범서비스에 들어가게 된다.

-한국전기통신공사
광역 데이터통신망 구축키로

한국전기통신공사는 90년대초에 본격적으로 상용서비스될 유선TV (CATV) 방송이 민간기업에 개방되는 것에 대비하여, 내년부터 전용케이블을 가설할 때 CATV 외에 신문지면, 사진등도 전송할 수 있는 대도시 광역 데이터통신망(MAN)을 91년까지 구축키로 했다.

CATV망이 가지고 있는 광역 전송능력을 이용하는 이 통신망은 4MHz 대의 주파수를 몇개로 나누어 고속으로 전송하는 방식으로 케이블을 통하여 음성데이터 영상정보의 쌍방향 교환이 가능하다.

통신망의 구성은 서울 부산등 대도시 지역의 전화국을 중심으로 하여 구축하면서 2천년대초까지 ISDN(종합정보통신망)의 전국적인 서비스 실시가 가능하도록 하는 것과 관련하여 이 망과 연계 운용되도록 추진키로 했다.

전기통신공사는 대도시 광역 데이터통신망을 구축하면서 초기에는 경제성을 고려해서, 동축 케이블을 가설한 뒤에 단계별로 광케이블로 전환시킬 방침이다.

-한국전기통신공사
ISDN 4개부문으로 구성키로

한국전기통신공사는 오는 94년에 부분적으로 실시될 ISDN(종합정보통신망)의 초기 통신망구조를 회선교환망, 전화망, 패킷교환망, 텔렉스망 등으로 크게 4개 부분으로 나누어 구성키로 했다.

동사에서는 94년부터 서울·부산 등 대도시 지역에서 실시할 초기단계의 ISDN은 •팩시밀리, 데이터단말기 등을 전송해 줄 회선교환망 •사설교환기, LAN(근거리통신망), 컴퓨터 등을 연결할 전화망 •텔렉스에 워드프로세서 기능을 추가한 텔렉스 및 데이터단말기 등을 접속시키는 패킷교환망 •텔렉스와 컴퓨터를 연계해 주는 텔렉스망으로 각각 구성 운용할 계획이다.

또한 동사는 연동장치인 IWU로 상호 연결될 이 통신망의 효율적 관리운용을 위해 대도시 교환시설은 지난 88년에 17%, 89년 21%, 90년 27%, 91년 32% 등으로 점차 디지탈화하고 전송시설도 시내전송로의 경우 88년 60%, 89년 65%, 90년 69%, 91년에는 73% 등으로 디지탈화를 점차 높여 나가기로 했다.

-금성반도체
통신용 에뮬레이터보드 개발

금성반도체는 업무용 퍼스컴에 내장해 IBM과 히다찌 대형 컴퓨터의 인텔리전트 단말기능을 수행할 수 있는 통신용 소프트웨어인 에뮬레이터보드를 개발, 시판에 나섰다.

동 소프트웨어는 서로 다른 기종의 컴퓨터나 통신기기끼리 정보를 송수신할 수 있으며, 퍼스컴에 내장될 경우 퍼스컴 고유기능수행은 물론 IBM, 히다찌 대형컴퓨터와의 고속 화일 송·수신도 가능해 퍼스컴과 이들 두기종의 대형컴퓨터를 접속할 때 별도의 단말기를 구입할 필요가 없다.

동사는 이번에 에뮬레이터보드와 함께 이를 운영할 수 있는 소프트웨어도 개발했는데 소프트웨어를 포함한 보드의 가격은 대당 50만원선이다.

-금성반도체
선로절체보드 시험장치 국산화

금성반도체(대표:이만용)가 선로절체 보드 시험장치를 국산화하는데 성공하였다.

그동안 선로절체보드의 시험장치가 국내에서는 개발되지 않아 전송로에 이상이 생기면 수작업으로 예비전송로를 교체해야 하는 불편을 겪어 왔다.

그러나 비음성 데이터를 고속으로 전송해 주는 가입자 반송장치에 내장하여 통신로의 소통을 원활히 해주는 기능을 가지는 선로절체보드의 시험장치가 개발되어 선로절체보드의 사전점검을 해주어 가입자 반송장치를 유지보수하는데 드는 비용을 절감할 수 있게 됐다.

-금성반도체
ATT와 전품목 기술협정 체결

지난 80년 금성과 미국 ATT사의 합작사로 설립된 금성반도체는 그동안 ATT사로부터 통신품목별로 기술도입계약을 맺고 이를 국내생산해 왔으나 최근 포괄적인 기술협력 계약을 체결 ATT사에서 생산되는 전품목에 대해 기술협력과 상호 국제분업체제를 구축하기로 했다.

양사가 체결한 기술협정을 보면, •전자교환기 등 제품개발 분야를 비롯

해 ●ISDN(종합정보통신망) 첨단 기술분야 ●통신망 등 네트워크 분야에 대한 상호기술협력과 기술 및 지원인력의 상호교류, 국내외 판매활동의 협력체제강화 등이다.

또한 양사는 이같은 기술협정을 효율적으로 추진키 위해 분기별로 「추진실무위원회」를 구성하기로 했다.

한편 금성반도체는 동 협정을 계기로 앞으로는 동남아, 중남미를 비롯, 최근 시장개척에 나선 중국, 동구공산권 등에 자사 상표로 수출에 나설 계획이다.

–금성산전

국내 최초 국산 교통관제시스템 설치 완료

금성산전(대표:이희종)은 최근 도심내 차량의 흐름을 효율적으로 제어하기 위한 중앙집중제어식 종합 교통관제시스템을 완전 국산화하여 대구시에 설치 완공함으로써 대구시 1차 순환선내의 교통정체 해소, 교통사고 및 차량공해 방지, 에너지 절약 등 도시의 교통난 및 교통공해 해소에 크게 기여할 수 있게 되었다.

이번에 설치가 완료되어 본격 가동에 들어간 금성산전의 교통관제시스템은 교차로 간격 및 신호표시방식 등 국내의 도로 및 교통제어실정을 고려, 신호주기의 조절 등이 가능하도록 한 한국형 종합 교통관제시스템이다.

동 시스템의 기능은 교통정보의 수집, 분석처리, 전달 및 제어기능으로 크게 나뉘어 지는데 시내 각 교차로에 설치된 차량감지기, 교통감지용 CCTV 카메라 및 교통 순찰차량 등으로부터 각종 교통정보가 수집되어 중앙관제센터의 컴퓨터에 자동송신되면 교통관제 센터의 컴퓨터에서는 입수된 모든 교통정보를 신속하게 분석, 처리하여 각 교차로 및 노선등을 포함하는 전 교통관제 지역의 교통신호 제어에 적합한 최적의 제어패턴및 파라미터를 선택, 결정하여 각 교차로 지역장치에 전달하게 된다. 이렇게 전달된 제어내용에 따라 최종전달 장치인 교통신호 제어기는 교차로 신호등을 자동으로 조정 제어하게 되며, 이때 이 제어내용은 관제센터의 교통상황표시판에 나타나게 되므로 방송국요원이 이 교통정보를 안내 방송하게도 된다.

금성산전은 동 국산 모델을 우리실정에 맞는 최적의 시스템으로 더욱 보완, 발전시킴은 물론 시스템을 표준화하여 전국의 대도시로 확산 설치할 계획이며 나아가 국내에서 축적, 토착화한 기술을 응용하여 우리의 사회적, 경제적 사정과 비슷하고 교통사정이 유사한 동남아 지역으로의 수출로를 확보 시장확대를 통한 외하 획득에도 일익을 담당할 방침이다.

–금성하니웰

CCTV방범시스템 국산화

금성하니웰(대표:홍성재)은 최근 국가 주요시설물 또는 산업현장의 보안, 감시를 위한 CCTV방범시스템의 주요부분인 고성능 CCD카메라와 모니터링 시스템의 국산화에 성공하였다.

동시스템은 세계 최첨단의 멀티시그널 영상전송방식을 채용하여 16대의 카메라 영상신호를 1개의 선으로 전송할 수 있으며 화면을 4개로 나누어 볼 수 있게 되어 있다.

또한 카메라 128대와 모니터 32대를 연결하여 128개 지역(최대 1024개 지역)의 침입자를 감시할 수 있으며, 퍼스널컴퓨터와의 폭넓은 인터페이스로 통합감시상황의 일보, 월보작성 및 카메라 경보상태감시 등의 각종 자료작성이 가능하며 경보 발생지역의 카메라 영상신호 자동녹화도 가능하다.

한편 동사의 CCTV방범시스템은 국가 주요 시설물이나 군사시설, 방송국, 발전소, 산업시설, 은행, 빌딩, 호텔, 쇼핑센터 등 사람이 많이 왕래하는 곳에 폭넓게 활용될 것으로 예상된다.

– 나라계산

빌딩자동화시스템 개발

나라계산은 컴퓨터로 빌딩의 환경과 설비상태를 자동으로 측정하고 전력·조명·방화·방범장비 등을 원격 제어할 수 있는 빌딩자동화 시스템인 MEGA 시리즈를 개발하여 보급에 나섰다.

중앙관제장치와 원격제어장치, 메가네트 등으로 구성된 동시스템은 중앙관제장치로 16비트 퍼스컴 AT기종과 32비트 퍼스컴이 사용되며, 원격제어장치는 프로그래머블 콘트롤러기능을 갖춘 것으로 2백51대까지 접속, 사용할 수 있다. 또한 메가네트는 원격제어장치를 상호 연결하기 위해 사용된 것으로 잡음의 영향을 적게 받아 시스템의 신뢰도를 높여 준다.

MEGA시리즈는 중소형 빌딩은 물론 대형빌딩과 공장제어에도 사용할 수 있으며, 특히 운용이 간편하고 확장성이 뛰어나 빌딩자동화를 추진하는데 적합하다.

빌딩자동화 시스템은 대부분 수입품으로 한글·한자 지원이 불가능하고 가격 또한 고가였는데, 동시스템의 개발에 따라 이러한 점들이 해소되었다.

–대우전자

교육용 컴퓨터 LAN 보급

올해부터 본격적으로 실시될 초·중·고교 컴퓨터 교육에 필수적인 LAN(근거리통신망)이 작년 하반기에 대우전자(대표:김용원)에 의해 개발, 본격적인 공급을 준비하고 있다.

MSX 방식의 8비트 퍼스컴을 연결하여 네트워크를 구성한 교육용 MSX LAN 시스템으로서 교사용 컴퓨터와 학생용 컴퓨터로 구성, 합리적인 정보의 전달, 교환, 프로그램의 송수신을 할 수 있으며 보조기억장치와 프린터를 공유할 수 있다.

또한 동 시스템은 최대 64대까지의 8비트 퍼스컴을 연결할 수 있으며 4개의 그룹단위로 나뉘어 화일서버를 장

치했을 뿐만 아니라 프린터에 메모리 기능을 부가하여 순차적인 출력이 가능하도록 하여 학습능률을 향상시킬 수 있다는 것이 특징이다.

– 대우중공업

로보트산업 강화키로

대우중공업(대표:이경훈)은 세계적인 로보트 전문업체인 일본 파낙사와 기술제휴를 맺고 로보트사업을 강화한다.

지난 12월5일 대우중공업의 이경훈 사장과 일본 파낙사의 이나바사장이 기술도입 계약서에 서명을 함으로써 이루어진 동 기술도입 계약에서는 스폿용접 및 아크용접 2개 기종 등 총 3개 기종의 기술을 도입하기로 약정하였다.

동사는 이에 따라 한국 파낙사로부터 콘트롤러부품, 일본 파낙사로부터 제조기술을 이전받아 로보트 본체를 생산키로 했는데 산업용로보트의 양산체제를 갖추기 위해 1백80억원을 투입, 창원에 FA공장을 건설할 계획이다. 창원공장의 로보트 생산시설의 규모는 1차년도 3백대로 예정되고 있다.

–대영전자

광전송 시스템 개발

대영전자공업(대표:정대영)에서는 음성과 데이터를 고속으로 전송할 수 있는 광전송 시스템을 개발, 공급에 들어 갔다.

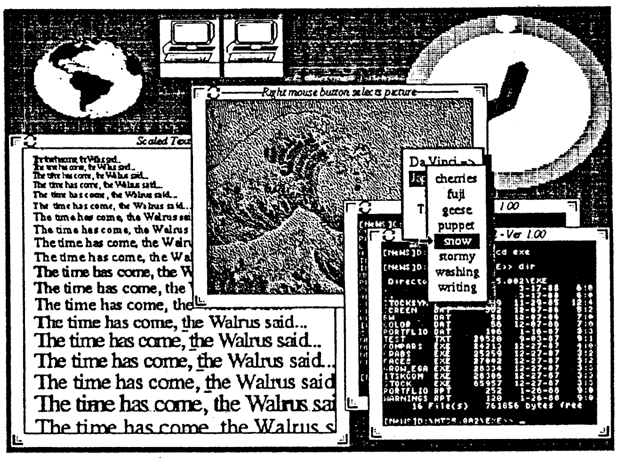
동시스템은 LSI(대규모집적회로)를 사용하여 채널당 107W 정도의 전력소모가 되며 최신 루프백 고장격리기능을 갖추고 있다.

또한 음성회선기준 2천16회선 용량을 1백39Mbps 속도로 전송할 수 있도록 설계하였다.

특히 3개의 DS3 신호를 다중화하여 3개의 채널을 접속할 수 있고 갑작스런 고정에 대비, 1대 11의 구간 보호설치기능도 지니고 있다.

–라이프모트사

국내 S/W시장 진출위해 (주)라이프모트 설립키로



홍콩에 본사를 두고있는 라이프모트사가 국내에 진출하여 (주)라이프모트를 설립할 예정이다. 현재는 소프트하우스인 MSI에 연락사무소를 설치해두고 회사설립에 필요한 준비활동과 홍보 및 마케팅에 필요한 정보를 수집하고 라이프모트사가 취급하는 소프트웨어와 관련된 하드웨어의 사용자와 시장을 조사해오고 있다.

앞으로 (주)라이프모트는 Software Publisher로서 국내에 필요한 소프트웨어 툴이나 유틸리티등을 공급하여 라이프모트 그룹과 유기적인 관계를 유지하면서 한국의 소프트웨어산업에 적극적으로 참여할 방침이다. 동사는 우선적으로는 이미지 소프트사의 NeWS/2와 Programmer'shop의 다양한 아이템들을 판매할 예정이다.

– 대원정보시스템

페이지프린터 개발

대원정보시스템(대표:이원우)은 B4 크기를 A4 크기로 축소 인쇄하는 기능을 갖고 있으면서 온라인 시스템에 적용될 경우 고출력 터미널용 프린터로 대체 사용할 수 있는 한글·한자 페이지프린터를 개발, 시판하고 있다.

이 프린터는 LED광에 의한 건식 전자사진방식을 채택하여 기존의 도트매트릭스 프린팅방식의 취약점인 헤드의 잦은 고장과 소음을 최소화 했으며, 연속출력시 A4 크기의 문서를 1분에 8매 인쇄할 수 있는 속도를 갖고 있다.

또한 일반 잡지 문자 수준인 2백40 DPI(1인치당 도트 수)의 고해상 인쇄가 가능, 사내 출판시스템으로 사용될 수도 있으며, IBM 5540 시리즈 퍼스컴에 접속하여 사용할 수 있다.

–동서컴퓨터

텔레비디오와 에이전트 계약

동서컴퓨터(대표:전홍철)는 지난 12월 3일 용산 전자랜드 본관에서 컴퓨터 다국적 브랜드인 텔레비디오사와 국내 에이전트 계약을 체결하고 계약식을 가졌다. 한편 이 자리에서 동사는 그간 회사의 기술개발과 소프트웨어부분 등에 기여한 한양대학교 전산연구소 임인칠박사와 경원대학교 김홍진교수에게 감사패 증정식도 가졌다.

동사는 그간 국내 유명메이커에 OEM방식으로 컴퓨터를 납품, 그 우수성을 인정받아 왔으며 이번에 텔레비디오사와 상호협의 끝에 계약을 체결하였는데 동사는 이번 계약을 계기로 텔레비디오사의 전품목을 용산 전자랜드 본관에 새로 마련한 동사 전시판매장과 종로매장에서 취급하며 이를 새로운

도약의 발판으로 삼아 사업의 규모를 확장할 예정이다.

또한 동사는 기존의 컴퓨터생산은 물론 컴퓨터 관련산업인 OA, FA, 컴퓨터그래픽, 내부모델 등의 사업을 강화할 방침이다.

-(주)마이크로소프트
한글 MS-DOS 시판

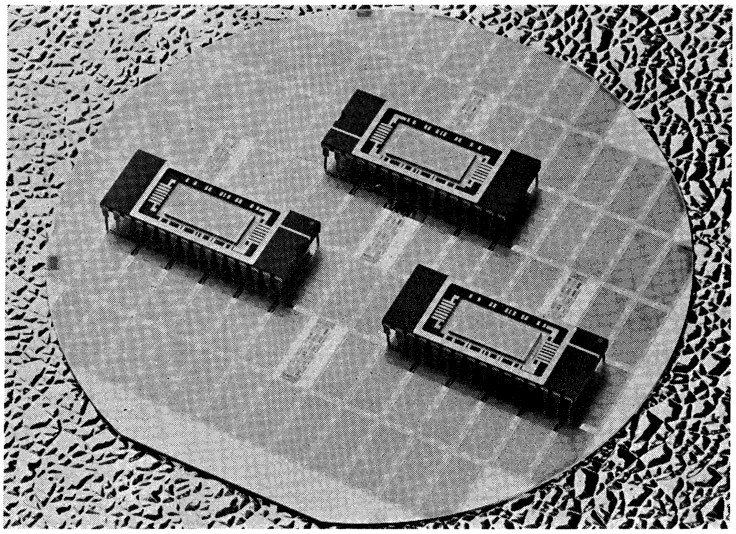
마이크로소프트(대표:이범천)는 16비트퍼스컴용 OS(운영체제)인 MS-DOS에 KS 한글표준인 2바이트 완성형코드를 채택하여 패키지 판매한다.

동 OS는 16비트 IBM 호환기종에 필수적으로 탑재하여 그동안 라이센스 계약에 의해 금성, 대우, 삼보등 10여 대형퍼스컴업체에만 공급해 왔다.

동사에서는 또한 DOS를 패키지 판매키로 결정함에 따라 소프트웨어의 불법복제방지 효과는 물론 중소퍼스컴업체의 영업 활성화와 응용소프트웨어의 패키지 판매유도에도 도움을 줄 것으로 기대되고 있다.

한글 MS-DOS는 프로그램 내용을 담은 디스켓 3매와 매뉴얼 2권으로 구성 판매된다.

-삼성전자
자체설계로 1M SRAM 개발



삼성전자(대표:강진구)는 지난 11월 16일 자체기술로 최첨단 반도체인 1MSRAM 개발에 성공하여 시제품 생산에 들어 갔다.

동제품은 256K SRAM보다 4배나 높은 집적도를 가지며 손톱만한 크기의 칩속에 6백여만개의 트랜지스터와 저항을 집적시킨 반도체로써 보통 신문지 8페이지 분량에 해당하는 약 13만자를 기억시킬 수 있어 제품의 성능을 크게 향상시킬 수 있으며, 휴대용 컴퓨터, 키폰, 다기능 전화기, 터미널, 팩시밀리, 각종 컴퓨터, OA기기, 정보통신기기, 군사기기 등에 폭넓게 적용된다.

이번 동사가 개발한 1M SRAM은 세계 최신의 공정기술인 다층배선구조를 이용하였으며, 기존의 제품보다 성능이 개선된 것으로 정보처리속도가 70나노초의 초고속, 저소비전력의 장점을 지닌 초집적 CMOS 제품으로서 앞으로 기술적 파급효과가 클 것으로 기대된다.

-쌍용컴퓨터
음성 자동응답시스템 개발

(주)쌍용컴퓨터 맨·머신 인터페이스의 복합적인 신호처리 기술을 요하는 컴퓨터 통신시스템중의 한가지로서, 정보를 필요로 하는 고객에게 공중 전화망(PSTN)을 통하여 각종 정보가 들어있는 호스트컴퓨터와 데이터 베이스로부터 고객이 요구하는 정보를 자연언어인 음성으로 제공할 수 있는 대화식 정보 입출력시스템인 VINUS-I (Voice Informatin Network for User Service)을 개발했다.

동제품은 소형 컴퓨터를 이용하여 시스템이 구성되므로 가격이 저렴하며 사용자의 입장에서 시스템을 구성하였으므로 컴퓨터에 대한 사전지식이 없어도 쉽게 사용이 가능하다.

또한 사용자의 여러가지 요구를 충족시킬 수 있는 음성편집, 합성기능, 전화사서함기능, 전화데이터 엔트리기능, 교환회선기능, 음성인식기능, 정보전달기능, 진단기 등 다양한 기능을 갖추고 있다. 시스템 운용상태가 CRT 및 스피커에 의해 모니터링되므로 신속, 정확한 유지보수 기능이 부여되며 신속하고도 지속적인 AS 및 기술지원이 가능하다.

-쌍용컴퓨터
신민·제일상호신용금고 전산화 추진

신민상호신용금고와 제일상호신용금고는 최근 경영효율화 및 고객에 대한 서비스 강화를 위해 쌍용컴퓨터가 개발하여 신용금고업계로부터 호평을 받고 있는 신용금고 종합관리시스템(SAMS)을 도입, 전산화를 추진키로 했다.

신민·대구제일상호신용금고는 이 시스템을 통해 여수신관리 등 창구업무로부터 고객관리, 인사급여관리, 경영정보관리 등 일반관리업무에 이르기까지 제반 업무를 자동화시켜 내부적으로 경영의 효율성을 높이고 고객에게는 신속, 정확한 서비스를 제공할 계획이다.

이번 양 신용금고에 설치키로 한 쌍

용컴퓨터(대표:최병항)의 「SAMS」는 자금관리, 이익율분석, 영업속보, 통계자료산출 등과 같은 각종 경영정보전달기능이 보강되어 경영합리화 및 고객의 신뢰성이 크게 향상될 것으로 보인다.

H/W로 신민은 HPM 3000/XE를 대구제일은 HP3000/925LX를 사용한다. 한편 신민·대구제일상호신용금고는 이번 업무전산화를 경영실적의 향상은 물론 쌍용의 교육센터를 이용자체 전산전문인력 양성의 계기로 삼을 방침이다.

-삼성HP
설계자동화 세미나 개최

삼성HP는 작년 11월 17일과 18일 양일간에 걸쳐"설계자동화의 발전 및 추세"에 관한 세미나를 조선호텔 1층 그랜드볼룸에서 가졌다.

아직 초기단계에 있는 국내 CAD분야 및 업계에 새로운 인식을 심어주고자 개최된 이번 세미나에서는 설계자동화의 발전과 추세에 관한 세미나에 이어 HP사의 설계자동화 추진사례도 전시됐는데, 전기전자 회로 및 PCB설계시스템(EDS/PC DS), 아날로그회로 및 PCB 설계시스템(AWB/DBS) 등의 EE-CAD 제품들과 기구설계시스템(ME 10/30), 유한요소 해석 패키지(NISA-II/Display-II)등의 ME-CAD 제품들이 선보였다.

이날 세미나를 담당한 HP사의 아시아 태평양지역 테크니컬시스템 마켓개발 매니저(시장개발 담당 매니저)인 John Schmidt씨는 CAD를 현업에 어떻게 도입하느냐하는 프로세서도입(단계적도입)의 중요성을 역설하기도 했다.

한편 SHP는 국내에서 최초로 CAD 컨설팅을 시도하고 있는 것으로 알려졌다.



-서울컴퓨터
여행사 업무관리 S/W 개발

서울컴퓨터(대표:김의균)는 여행사의 여행 및 거래처 관련 업무를 효율적으로 관리할 수 있는 여행사 업무관리 종합시스템을 개발, 시판에 나섰다.

동 소프트웨어는 퍼스컴용으로 여행사의 인사급여·회계·판매업무를 자동처리할 수 있을뿐 아니라 고객 및 거래처를 업종별로 분류 검색할 수 있어 대량의 우편물 발송에도 손쉽게 활용할 수 있다.

또한 마이크로컴퓨터용 프로그램언어인 RM코볼로 설계되어 있어 데이터 공유 처리가 가능해 부서 및 여행사간의 데이터 교환이 수월하며 업무성격에 맞게 프로그램을 쉽게 변환할 수 있다.

-엘레컴엔지니어링
미 ADT사와 국내 독점 판매계약 맺어

PCB회로 설계용 CAD/CAM 시스템을 전문적으로 공급하고 있는 (주)엘레켐엔지니어링(대표:이완)은 최근 아날로그 회로설계 및 분석전용 S/W 분야의 세계적인 공급전문업체인 미국 Analog Design Tools(ADT)사와 국내독점 판매계약을 맺고 시판에 나섰다.

아날로그회로 설계 및 분석전용 S/W인 Analog Workbench(AWB)와 PC Workbench는 회로 설계자를 위하여 개발된 분석전용 시스템으로서 무제한의 스키메틱 디자인이 가능하며, Spiceplus를 이용, 각종 테스트 인스트루먼트 시뮬레이션을 할 수 있다.

특히 풍부한 라이브러리와 회로의 워스트 케이스 결과를 산출해 내는 분석모듈, 비선형 자화 시뮬레이션이 가능한 파워디자인모듈, 신뢰성 시뮬레이션이 가능한 스모크 알람모듈등이 다양하게 제공되어 있으므로 아날로그 회로설계의 최적화 및 고신뢰성을 도모할 수 있는 최첨단 S/W이다.

또한 현재 국내에서 많이 보급되어 가동중에 있는 유닉스 베이스의 각종 첨단 엔지니어링 워크스테이션(HP, Sun, Apollo) 등에서 쉽게 시스템 구축이 가능하여 국내 공공기업체 및 연구소에서 수요가 급증할 것으로 예상된다.

-왕컴퓨터코리아
통합화상 정보관리시스템 시판

왕컴퓨터코리아에서는 문자영상음성 등 모든 형태의 정보를 종합적으로 처리해 주는 통합화상 정보관리시스템(WIIS)을 공급한다.

동시스템은 광디스크 마이크로필림, 마그네틱 디스크 등 다양한 기록매체를 이용해 각종 정보를 데이터베이스로 관리, 필요할 때 손쉽게 검색하고 송수신할 수 있는 사무자동화(OA)용 종합 패키지로서 왕오피스 전자우편을 통해 컴퓨터통신을 할 수 있으며 팩시밀리와도 연결하여 원본서류를 그대로 송수신할 수 있다.

또한 메뉴선택 방식으로 설계되어 있어 사용에 편리하다.

한편 왕컴퓨터코리아는 동 시스템을 중형컴퓨터인 VS500 시리즈와 VS7000 시리즈에 각각 전자화일형과 네트워크용으로 시스템을 구성, 판매에 나섰다.

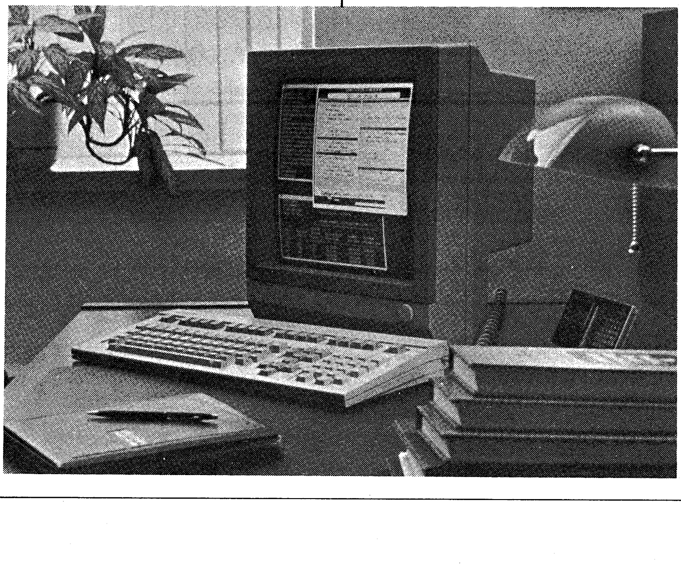
-왕코리아

코오롱상사와 WIIS 판매대리점 계약 체결

왕컴퓨터코리아(주)는 최근 (주)코오롱상사와 자사의 통합화상관리 시스템(WIIS)의 판매 대리점 계약을 체결했다. (주)코오롱 상사는 Wang의 VAR (Value Added Reseller)로서 시스템 판매와 함께 고객요구에 따른 분야별 어플리케이션도 개발, 지원할 예정이다.

WIIS는 문서, 데이터, 화상 등 모든 형태의 정보를 보관, 관리, 검색할 수 있고 온라인 전송이 가능한 새로운 개념의 시스템으로 데이터 프로세싱 및 커뮤니케이션 기능이 이미지 프로세싱과 통합되어 있는 것이 특징이다.

WIIS는 Wang VS 시스템과 4세대 언어 어플리케이션 개발 및 데이터베이스 관리시스템인 PACE를 기반으로 하는 시스템 패키지로 서명된 계약서, 고객 및 환자기록카드, 어음, 증권, 신문 등 모든 화상 형태의 정보를 포함한 문서나 데이터를 자기 디스크, 광디스크, 또는 마이크로 필름에 보관하고, 필요시 워크스테이션에서 윈도우를 이용하여 다중디스플레이를 할 수 있으며, 색인을 통하여 손쉽게 관리할 수 있다. 또 레이저 프린터를 이용하여 원본을 그대로 재생산할 수도 있다.



-제일전산

종합정보시스템 시판

제일전산(대표:박태규)은 최근 하나의 시스템으로 음성정보서비스·홈 트레이드 및 객장용 정보조회 터미널을 동시에 지원할 수 있는 종합정보시스템(TISS)을 개발, 시판에 나섰다.

이 시스템은 하나의 DB를 이용, 이같은 서비스를 함께 제공할 수 있을 뿐 아니라 하나의 회선을 통해 음성 및 문자정보를 통합 전송할 수 있어 네트워크관리가 간편하다는 특징을 가지고 있다.

또한 이 시스템은 32비트 마이크로 컴퓨터「시스템 3200」을 주축으로 구성돼 최대 32회선의 음성 및 문자정보를 제공하고 있다.

설치비용이 시스템당 5천만~1억원 수준으로 각 서비스별로 별도로 운용하는 것보다 경제적이며, 주요 지점에 설치할 경우 각 지점 단위에서 업무의 상당부분을 자체 소화할 수 있어 호스트의 부하를 줄일 수 있다.

제일전산측은 증권사와 신속하고 다양한 정보가 요구되는 고속도로 안내, 일기예보 등 각종 분야에도 효과적으로 쓰일 수 있을 것으로 기대하고 있다.

-한국데이타통신

MHS시험서비스 실시

데이콤은 금년 4월 퍼스널컴퓨터와 텔렉스간의 본격적인 통신서비스를 시작하기에 앞서 지난해 12월부터 국내 20개 기관의 퍼스컴과 텔렉스를 연결해 통신을 주고 받는 시험에 들어 갔다. 또한 퍼스컴과 텔렉스를 팩시밀리와 연결시켜 주는 서비스도 금년 가을부터 선보일 예정이다.

MHS소프트웨어는 국제 통신표준기관인 CCITT의 통신규약에 따라 개발하였으므로 외국과의 통신도 가능하다.

현재까지 시험서비스 참여를 희망한 기관은 두산산업, 한일은행, 동남해운, 대우자동차부품 등이다.

-한국IBM

고속 토큰링 네트워크 발표

한국IBM은 진보적인 응용과 근거리 통신은 물론, 광역의 통신망을 효율적으로 구축할 수 있는 고속의 토큰링 네트워크(Token-Ring Network)를 발표했다.

동시스템은 16Mbps와 4Mbps의 네트웨크에 모두 사용이 가능하며 또한 워크스테이션, 통신제어장치, 중소형 컴퓨터에 직접 접속시킬 수 있다.

또한 16Mbps와 4Mbps에서 겸용이 가능한 토큰링 브리지(Token-Ring

Bridge) 프로그램으로서 원격지 통신을 위한 브리지기능도 수행할 수 있다.

한편 새로 발표한 동제품으로 인하여 IBM사는 토큰링 네트웍크의 데이터 전송속도를 향상시키게 되었고 사용자들도 토큰링 네트웍크에 진보적인 그래픽 및 영상처리용 응용프로그램을 적용시킬 수 있을 뿐만 아니라 과학기술응용 분야에 있어서도 높은 속도와 성능을 보장받게 되었다.

-한국IBM(주)
수출 1억불탑수상기념 리셉션

한국 IBM(주) (대표:피터 로울리)는 지난 12월 6일 수출의 날 1억불탑 수상기념 리셉션을 영등포구 여의도동 구내식당에서 가졌다.

이날 리셉션에는 수출 1억불탑 수상 축하를 위해 중소기업 및 대기업 컴퓨터 부품공급업체의 간부 100여명과 한국 IBM 직원등 약 150여명이 참석, 한국 IBM측은 컴퓨터부품을 공급해 온 한국전장주식회사와 대우전자주식회사 등 30여개사의 중소기업 및 대기업 간부들에게 감사패를 증정했다.

한국 IBM은 이번으로 연속 2회의 수출 1억불을 초과 달성한 것으로 알려졌다.

-한국IBM
의료정보 시스템 개발

한국IBM은 외래, 입원환자의 등록과 접수업무를 비롯한 환자의 진료정보조회, 처방등록, 진료비 계산, 수납 보험료청구 등의 업무를 자동으로 처리할 수 있게한 환자관리 및 회계 시스템과 의사의 처방결과와 그에 따른 처리, 메시지 전송등의 기능을 가진 처방전달 시스템을 개발하였다.

한국IBM의 프로그래밍 센터와 연세대 세브란스 병원팀이 약 2년에 걸쳐 공동개발한 이 시스템은 IBM의 중·대형 컴퓨터를 이용한 병원업무 전산화용 소프트웨어로 효율적인 병상운영을 지원할 수 있다.

-한국무역정보통신
한글컴퓨터 통신시스템 개발

일반 전화회선을 이용한 한글컴퓨터 통신시스템이 한국무역정보통신(대표:주영훈)에서 개발되어 지난 11월 22일 한·미간 첫 시범운용을 가졌다.

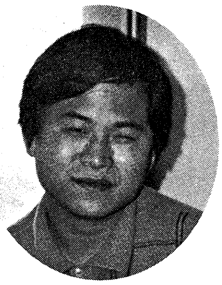
그동안 전화회선을 통한 컴퓨터통신이 여러차례 시도되어 왔으나 한글을 이용한 것은 이번이 처음이다.

동 시스템은 대한무역진흥공사(KOTRA) 서울본사와 미국 마이애미지사의 컴퓨터를 전화선으로 연결하여, 미국에서 검색이 가능한 전자게시판 기능을 가지고 있으며 KS 한글표준코드를 채택한 것이 특징이다.

한국무역정보통신은 KS 한글전자사서함과 한글다중방송 시스템도 발표할 계획이다.

□ 미니인터뷰 / 이우관

시각장애란 사실이 컴퓨터를 배우는데 장애가 되지는 않는다고



"무엇보다도 시각장애자를 위한 도구(한글인식이 가능한 점자프린터나 신서사이저, 음성합성, 음성인식장치 등)의 개발이 국내에서도 활발히 이루어져야 시각장애자들에게도 컴퓨터가 확산될 것입니다."라고 말하는 이우관씨.

그는 시각장애자이면서 컴퓨터(그가 소유한 기종은 IBM-PC/AT기종이다)에 스피커라이저(신서사이저의 일종)를 장착하여 사용하고 있는 컴퓨터광이다.

서울맹아학교(12년 과정)를 거쳐 대구대학 특수교육학과를 졸업한 그는 대학 재학시절, "사회교육 프로그램"이라는 강의를 듣고 컴퓨터에 관심을 갖기 시작했다고.

그는 직장(한국맹인복지협회)에서는 문서작성이나 관리 등에, 개인적으로는 언어학습(파스칼, 베이직, dBASEⅢ, 어셈블리어 등)이나 번역, 가계부작성 등에 컴퓨터를 주로 활용하고 있다.

맹아학교 4학년때부터 익혀온 타자실력으로 컴퓨터의 키보드를 사용하는데는 별 문제가 없지만 문서작성이나 프로그래밍시에는 주위사람이 일일이 읽어줘야 하며, 에러가 발생한 경우에는 스피커라이저의 에러메시지를 듣고 수정하여 실행시킨다고. 컴퓨터 관련 참고도서는 점자번역(점역)을 하여 참고하고 있다고 한다.

이러한 일련의 작업들이 힘들고 어렵지만 그에게 있어 시각장애는 컴퓨터를 익히는데 아무런 장애가 되지 않는 듯하다. 오히려 그가 안타까워 하는 일은 많은 시각장애자들도 조금만 노력하면 남들과 똑같이 새로운 세계를 맛볼 수 있는데 그러지 못하는 것과 시각장애자를 위한 도구들이 없어 많은 시각장애자들이 아예 컴퓨터를 접할 수 없는것이라고.

컴퓨터 하드웨어쪽으로도 많은 관심을 보이는 이우관씨는 아직 국산화되어 있지 않는 시각장애자를 위한 컴퓨터관련 주변기기를 개발하는 것이 꿈이라고 한다.

1961년생으로 지난 88년 결혼한 그는 퇴근시간 이후의 거의 모든 시간을 컴퓨터와 보내는 덕택에 부인(권현미씨)의 밉지 않은 투정을 감수해야 하지만 자신에게 밝은 빛을 가져다 준 사람이라고 말하며 행복한 웃음을 짓는다.

국외소식

미, PC/AT 호환기메이커
IBM에 대응하여 새로운 버스「EISA」발표

최근 컴팩, AST리서치, 탠디 등 미국 퍼스널컴퓨터 메이커 9개사는 공동으로 32비트 퍼스컴용의 새로운 버스 아키텍처 EISA(Extended Industry Standard Architecture)를 발표하였다.

현재 널리 보급되어 있는 PC/AT 버스 (컴팩 등은 이것을 ISA(Industry Standard Architecture)라 한다)와 완전한 호환을 갖는 동시스템은 IBM PS/2의 시스템버스인 마이크로 채널(MCA)과 거의 동일한 기능, 성능을 갖고 있다.

컴팩 등 미국의 AT 호환기메이커 각사에서는 금년중에 EISA를 사용한 32비트 퍼스컴을 개발하여 IBM의 PS/2에 대항할 예정이다. 인텔이나 마이크로소프트사 외에 50개사가 넘는 호환기메이커, 주변기기메이커, 소프트하우스 등도 EISA에 찬성하고 그중에 PS/2 호환기메이커인 딜컴퓨터도 참여하고 있다. 89년이후 EISA규격의 PC/AT계 퍼스컴과 PS/2를 중심으로 한 MCA 퍼스컴간에 치열한 시장경쟁이 예상된다.

AT버스와 호환이 가능한 EISA

표 1에 EISA, ISA(AT버스), MCA 버스 아키텍처의 규격을 비교하였다. ISA의 단점을 개선하여 충분한 연구끝에 EISA가 개발되었다. EISA에서는 데이터전송 속도가 각단에 고속화되고 32비트 일괄 전송모드에서는 최고 33메가바이트(MB)/초의 전송능력을 갖는다.

어댑터(I/O)카드의 설정에 있어서도 EISA는 MCA의 POS(Programmable Option Select)기능과 동일한 기능을 소프트적으로 행한다. PBS(Program Board Setup)로 딥스위치나 점프선을 하드웨어적으로 조작하는 것이 아니고 자동적으로 시스템을 구성할 수 있다. 어드레스 버스도 80386 MPU 에 포함시켜 32비트화되어 사실상 메모리의 제약이 없어진다. 즉, EISA는 32비트 대응의 멀티태스크 기능, 멀티미디어 데이터처리를 의식한 규격으로 되어 있다.

EISA의 등장이유

EISA는 이와 같은 특징과 함께 PC/AT의 ISA버스와도 호환성을 갖는다. 자료에 의하면 ISA규격과 EISA규격의 I/O기기가 동일 시스템상에서 병행되어 동시에 움직인다고 한다.

컴팩 등이 EISA를 개발하게 된 것은 IBM의 MCA를 채용할 경우 많은 로얄티를 지불하지 않으면 안되기 때

〔표 1〕ISA(PC/AT 버스), 마이크로 채널, EISA의 규격 비교

	ISA(PC/AT) 버스	마이크로 채널	EISA 버스
제어신호	8288 버스 콘트롤러와 유사	80386/286 출력과 유사	ISA 버스의 수퍼세트
어드레스 폭	24비트	32비트	32비트
데이터 폭	16비트	16, 24, 32비트	8, 16, 32비트
DMA	7채널	15채널	7채널
버스 클럭	MPU에 존재	MPU와 독립	MPU와 독립, DMA 채널에 4개의 전송사이클을 설정
데이터전송속도	MPU에 존재	최고 20MB/초	최고 33MB/초
어댑터(I/O카드) 설정	스위치에 의해 설정	POS 기능에 의해 소프트적으로 행한다.	자동시스템 구성 기능에 의해 소프트적으로 행한다.
멀티 미디어	—	오디오, 비디오의 신호선 있음	—
비 고	MPU로의 부하, 기능 집중형 시스템쪽, 싱글 태스크 처리에 적합하지만 멀티태스크에서도 어느 정도의 처리는 가능, 코스트가 안정	기능 분산형 시스템쪽보다 큰 시스템의 1개의 구성요소로 퍼스컴을 사용할 경우에 적합하다. 코스트는 상대적으로 높다.	ISA와 호환성을 유지하고, 마이크로 채널의 기능을 의식하여 개발되었다.

문이다.

발표에 의하면 EISA버스를 적용한 제품의 등장은 89년 후반쯤으로 우선 미국의 인텔사에서 EISA용의 입출력 콘트롤러, 버스콘트롤러, 버스마스터(master)를 VLSI형으로 제품화하고 각 퍼스컴메이커에서는 이것을 채용할 예정이다. 그 후 주변기기 메이커나 소프트하우스 각사에서는 EISA관련 제품이 90년경에나 등장되리라 생각된다.

대책을 모색하는 IBM

이와같은 호환기메이커의 움직임에 대하여 IBM사에서는 대책을 세우고 있다. EISA의 침투를 막기 위해 89년 전반까지 MCA를 채용한 PS/2의 판매에 주력할—MCA의 로얄티를 인하하고, PS/2 호환기를 증가시켜 판매대수의 증가를 도모— 예정이다.

IBM은 EISA발표와 동시에 PS/2의 하위모델인 30/286을 발표하였다. 이것은 MCA보급에서는 역효과라는 소리가 높다.

일본
컴퓨터바이러스보험 등장

최근 일본에서는 컴퓨터바이러스에 대한 보험이 등장하여 관심을 모우고 있다.

손해보험업계에서는 컴퓨터종합보험 약관을 개정하여 컴퓨터바이러스로 인한 데이터프로그램 피해를 동 보험의 보상대상에 추가하기로 결정하였다.

일본에서 지난 75년부터 기업을 대상으로 판매해 오던 컴퓨터종합보험은 컴퓨터 본체, 즉 하드웨어의 사고에 국한되어 적용되어 왔는데 이번의 약관 개정은 컴퓨터바이러스의 침입으로 인한 정보와 프로그램 파괴의 경우, 본래의 상태대로 회복시키는데 필요한 비용을 보험회사가 보험금으로 지급하게 되는 것이다.

동 보험금은 시스템의 규모, 프로그램의 제작시간, 소멸된 데이터를 축적하는데 드는 비용을 감안해 산출하며 정보가치의 경우는 같은 정보라도 기업에 따라 중요성이 다르다는 이유로 제외됐다.

인텔 등 18개사
AT&T의 차세대 UNIX 지원 확고히 다져

90년 초쯤, 통합 UNIX시스템 최종 버전 발표

미·일·유럽의 컴퓨터, 소프트웨어 메이커 18개사는 UNIX의 표준규격화에 대해서 AT&T사의 차세대 UNIX 를 지원하는 입장을 새삼스럽게 확인함과 동시에 단체를 설립한다고 발표하였다.

UNIX의 표준규격화에 대해서는 AT&T 사와 Sun Microsystems사가 결성한 그룹과 IBM사와 DEC사 등이 결성한 OSF(Open Software Foundation) 등 2개의 그룹이 각각 표준화 작업을 진행하고 있다.

그중 AT&T그룹이 단결의 재확인을 한 것이다. 동그룹은 지난 87년 12월에 AT&T사와 Sun사가 공동으로 UNIX의 통합화를 진행한다고 발표한 이후 업계에 파문을 일으키고 있다.

이번에 AT&T그룹을 지원하고 나선 18개사중에는 후지쯔, 도시바외에 굴드사나 암달사, 콘트롤데이터사, 모토롤라사, 인텔사, 유니시스사(이상은 미국), 올리베티사(이태리) 등 큰 메이커가 상당수 포함되어 있다.

이들 18개사는 앞으로 UNIX System V 릴리스4.0에 의거하여 제품을 개발, 통일화 작업을 진행할 방침이다. 지난 88년 10월 중순에 통합 UNIX System V 릴리스4.0의 상세한 사항이 동경에서 공개되었는데, 그것에 의하면 소스코드는 β버전이 올해 제2/4분기, 최종버전이 제3/4분기에 선보일 예정이고, 90년초에는 바이너리코드의 최종버전 등 릴리스4.0관련의 소프트웨어가 대략적으로 모두 나올 것으로 예상되고 있다.

반면에 매사추세츠주 안드바에서 개최된「제1회 OSF멤버회의」에 의하면 OSF에 10단체가 가맹되었다고 한다.

OSF는 가맹 각 단체의 출자에 의해서 UNIX의 통일규격을 추진하는 것을 목적으로 한 산업계 최초의 조사, 개발조직이다. 앞으로 15~21개월동안 IBM/AIX를 기반으로 한 UNIX를 중심으로 확장이나 서브시스템을 포함하는 표준 UNIX를 개발할 계획이다.

이번 멤버회의에서는 주로 OSF 시스템의 표준그래픽스인터페이스의 후보에 있던 규격을 비교 검토하였다. 신규가맹단체는 88오픈콘서시엄사, AMD사 부즈에렌앤드해밀턴사, 마이컴인터런사, 놀스크테이터사, 패시픽벨사, 실리콘그래픽스사, 스탠포드대학, 스웨디시텔레콤그룹 및 Wang Laboratory 등 10개 단체이다.

OSF회장 헨리 그루스씨는 "신멤버는 연구, 컨설팅, 하드웨어, 시스템공급자 텔레커뮤니케이션 등 각 분야에서의 톱클래스 기관들로서 OSF는 그래픽 유저인터페이스의 선택에 착수하고 있으므로 이들 신규멤버의 가입은 상당히 고무적"이라고 AT&T진영을 견제하는 발언을 하였다.

아시아지역에서는 최초로 히다찌 참여

또한, 히타찌제작소는 일본기업으로서는 처음으로 OSF에 스폰서로서 참가하였다고 발표했다. 동사는 OSF의 9번째 스폰서이나 아시아·태평양지역의 기업으로서는 최초의 참여업체이다.

히다찌 이외의 스폰서기업은 아폴로컴퓨터사, 그루베블사, DEC사, 휴렛팩

커드사, IBM사, 닉스돌프사, 필립스사, 시멘스사 등이다.

치열한 접전 예상되어

양진영의 대립은 격화된 것 같으나 표면적으로는 서로 양보하여 의견을 일치시킬 것을 전제로 한 논의가 이루어지고 있는 듯하다.

AT&T UNIX Pacific사의 래리그룸사장은 "양진영은 통합화에 대해서 활발한 토의를 거듭하고 있다"고 말해 톱레벨에서는 논의가 이루어지고 있는 것으로 밝혀졌다.

AT&T사와 Sun사그룹의 UNIX system V 릴리스4.0이 모습을 나타내기 시작하고 있는데 비해 OSF 진영의 스펙은 전혀 밝혀지지 않고 있다.

양진영은 지난 10월 초순에 비공식적인 논의를 하였다. 회의의 내용은 X/OPEN사양에 의거하여 표준 UNIX의 임플리먼트(implement)를 한 것으로 전해지고 있으며, 그중에서 AT&T 진영은「AIX 대신에 System V 릴리스 4.0을 중심으로 해야 한다」고 하는 제안을 하였다. 이에 대해 OSF진영이 AIX에 릴리스4.0을 거두어들이는 안을 강조한 결과 논의는 평행선으로 끝났다고 한다. 양진영이 확인한 것은 UNIX 의 표준화작업에 있어서는 X/OPEN이 키포인트가 된다는 점이였다.

아폴로컴퓨터
MCAE용 소프트웨어 2종, 자사 신제품에 사용

아폴로 컴퓨터사는 MCAE(Mechanical Computer Aided Engineering) 분야의 우수한 응용 소프트웨어인 PDA 엔지니어링사의 Patran과 SDRC사의 I-DEAS가 아폴로의 신제품인 시리즈 10000퍼스널 수퍼컴퓨터에 운용된다고 발표했다.

이로써 PDA와 SDRC도 아폴로의 시리즈 10000을 자사의 제품라인에 통합하기로 한 50여개의 소프트웨어/하드웨어 공급업체에 속하게 되었는데, 이들 공급업체들은 현재 EDA(Electronic Design Automation)와 CAD/-CAM, 데이터베이스 매니지먼트, 매핑 및 이미징 등에 광범위한 시장을 구축하고 있다.

아폴로사는 복잡한 디자인에서 요구되는 강력한 모델링과 아날리시스 툴을 제공하는 Patran과 I-DEAS같은 MCAE 응용 소프트웨어에는 시리즈 10000이 가장 적합한 시스템이라고 한다. 또한 퍼스널 수퍼컴퓨터의 성능과 다양성을 엔지니어링 환경에 도입함으로써 최초로 모델링과 분석업무를 한꺼번에 처리할 수 있게 되었는데, 이는 예전의 크고 고가인 메인프레임에서 배치 프로세싱으로 수행되었던 작업이다.

금년 2/4분기에 시리즈 10000에 포팅(porting)될 예정인 SDRC사의 I-DEAS는 복잡한 기계제품의 디자인이나 분석, 시험에 사용되는 시스템이다. I-DEAS(Integrated Design Engineering Analysis Software)는 제품생산에 쓰이는 개발시간과 비용을 크게 줄여줄 뿐 아니라, 생산에 앞서 사용자들이 그 설계를 최대한 활용할 수 있게 해줌으로서 제품의 질을 높여주고 있다.

이 시스템은 금형이나 유한요소의 모델링, 분석, CAT, 제도, 생산에 필요한 통합된 소프트웨어 모듈을 제공함으로써 전제품의 개발과정을 완벽히 지원한다. 또한 I-DEAS는 오픈 아키텍처에 기초함으로 기존의 CAD/CAM 시스템들과 쉽게 인터페이스될 수 있으며, 사용자가 개발한 소프트웨어와의 통합도 가능하다.

SDRC는 아폴로의 NCS(Network Computing System)를 채택하고 있는데, NCS는 compute-intensive디자인과 분석업무를 네트워크를 통해 전문기종에 분산시키는 역할을 한다. 또한 아폴로 워크스테이션 네트워크를 사용하는 엔지니어들은 아폴로의 open dialogue 를 사용하여 여러 기종에 분산되어 있는 I-DEAS 소프트웨어를 쉽게 사용, 분배, 관리할 수 있다.

시리즈 10000 퍼스널 수퍼컴퓨터는 고체모델링이나 유한요소 해석과 같은 업무를 도와주며, 설계 및 엔지니어링, 테스팅 등 전과정에서 수퍼컴퓨터의 성능을 사용할 수 있게 해준다.

아폴로는 시리즈 10000 이외에 여러 모델을 선보이고 있다. 모토롤라의 MC 68030 마이크로프로세서를 사용하는 시리즈 3500과 시리즈 4500은 가격 대 성능비가 높다. 또 시리즈 3500은 저렴한 가격의 4MIPS 시스템이며, 시리즈 4500은 7MIPS 제품이다.

칩스사 아스키와 합작
랩탑용 VGA 호환 LSI개발

미국의 칩스 엔드 텍크놀로지사와 일본의 아스키사는 랩탑머신용의 VGA 호환 그래픽스 콘트롤러「82C455」를 개발, 샘플출하를 시작함과 동시에 칩스사는 동 콘트롤러와 조합시켜 고기능 랩탑머신을 구축할 수 있는 시스템 칩세트「LeAP」를 개발, 발표하였다.

82C455는 LCD나 프라즈마, EL이라는 랩탑 머신의 표시 패널을 직접 드라이브할 수 있는 VGA 대응의 콘트롤러로 표시 패널에 따라서 다른 각종 크기나 해상도에 대응하고 있으며 패널 인터페이스에 직접 접속할 수 있다. 컬러 패널에서는 64색까지 표시가능하다.

리프는 80286/80C286용과 386S×용의 2종류가 있다. 내역을 보면, ①CPU/버스/메모리 콘트롤러, ②버퍼/버스 컨버젼 콘트롤러, ③파워 콘트롤유니트, ④페리패럴 콘트롤러의 4개 칩이

다. 동 칩세트를 사용하면 시스템용, 그래픽용 BIOS를 1개의 EPROM에 내장할 수 있고 리프를 포함하여 부품수 32개(메모리는 제외)의 LSI로 PC/AT 호환의 랩탑머신을 구축할 수도 있다.

일본, ICOT
세계 최초로 5세대컴퓨터 발표

세계 최초로 일본에서 5세대컴퓨터의 시제품이 개발되었다.

신세대 컴퓨터 기술개발기구(ICOT)는 일본정부와 컴퓨터업계의 공동출자로 세워진 기구로서 최근 기억, 연산, 추론 등 인간의 사고능력에 가까운 기능을 가진 5세대 컴퓨터 1호기를 개발하였다.

동 컴퓨터 개발은 ICOT가 지난 82년부터 10년 계획으로 추진해 온 사업으로 모두 5백억엔 이상의 자금이 투입될 예정이다.

4세대 컴퓨터는 최대규모 집적회로를 사용하며, 프로그램된 정보를 기계적으로 처리, 한번에 하나의 일밖에 수행할 수 없었다.

반면 5세대 컴퓨터는 인간에 가까운 기능을 갖고 여러종류의 일을 동시에 처리할 수 있다.

새컴퓨터개발에 지도적역할을 맡고 있는 일본 통산성은 이번 1호기를 기초로 추론, 지식기능, 처리능력을 하드웨어·소프트웨어 양측면에서 대폭 향상시켜 계획이 끝나는 오는 91년말까지 실용화한다는 구상을 세워놓고 있다.

애플사
NeXT에 대응하여 Macintosh Ⅱx 발표

MacⅡx 는 Mac패밀리의 최상위모델로 지난 88년 8월에 개최된 Mac-World Expo에 등장한 MacⅡ 모델4/40의 기능을 강화시킨 시스템이다(기본규격은 표 2참조).

MacⅡx의 주요 강화점을 살펴보면

- CPU는 클럭주파수 15.7MHz의 68020과 동일한 클럭주파수를 갖는 68030을 채용하고 부동소수점 연산 코프로세서는 68882를 사용한다.
- 내장 플로피디스크 드라이브의 용량은 800k바이트에서 최대 1.44M바이트까지 확장 가능하다.
- ROM은 메인 메모리와 동일한 SIMM (Single In-line Memory Module) 보드에 탑재된다.
- 하드디스크의 용량은 40M바이트에서 최대 80M바이트까지 확장 가능하다.

등이다. 물론 MacⅡ 패밀리에 대응하는 응용 소프트웨어는 모두 동작한다.

내장 FDD도 MS-DOS나 OS/2상의 데이터에 대응

CPU와 함께 MacⅡx의 중요사항은 용량 1.44M바이트(720K바이트 포맷도 가능)의 3.5인치 FDD를 채용한 것이다. 이것은 현행의 800K바이트 FDD의 약 2배의 기억용량을 실현한 것이다. 동 디스크 드라이브의 주요 특징으로는

- 현행 Mac 패밀리의 400/800K 바이트 디스크의 리드/라이트가 가능.
- 3.5인치 베이스의 MS-DOS나 OS/2포맷의 데이터 리드/라이트가 가능하다.

또 MS-DOS나 OS/2로의 대응은 동사가 88년초에 밝힌 전략을 구체화한 것으로 Mac의 어플리케이션으로 작성한 데이터를 MS-DOS나 OS/2 대응의 어플리케이션용 데이터로서 리드/라이트할 수 있다. 예를 들면, Mac으로 작성한 EXCEL의 화일을 MS--DOS 상의 Lotus 1-2-3으로 사용된 wks 포맷의 화일로 변환하여 기억할 수 있다.

ROM의 탑재방식이 SIMM 방식으로 변경

SIMM은 현행의 Mac패밀리가 채용하고 있는 메인 메모리의 탑재방식으로 256k비트 DRAM칩을 8개 탑재한 소형보드를 4매 채용하여 1M 바이트의 메인 메모리를 실현하고 있다. 동 시스템에서는 이와 같은 방식을 시스템 ROM에도 채용하여 현행기종에서는 불가능한 ROM의 교환이나 증설이 가능하도록 하였다.

〔표 2〕 매킨토시 Ⅱx의 주요 사양

CPU	MC68030(클럭주파수 15.7MHz)
코프로세서	MC68882(클럭주파수 15.7MHz)
ROM	256K바이트(SIMM방식 보드에 탑재)
RAM	4M바이트(최대 8M바이트까지 확장 가능)
내장 디스크	플로피디스크 모델 : 1.44M바이트 FDD×2대 1.44M바이트 FDD×1대 하드디스크 모델 : 80M바이트 HDD×1대
인터페이스	RS-422C×2, SCSI 패러럴×1, ADB×2, 스테레오 사운드×1, NUBUS×6
사운드 제너레이터	4음성 스테레오 샘플링 제너레이터 내장

서클동정

건국대학교
제14회 프로그램 전시회 음성 응답기에 관심 집중

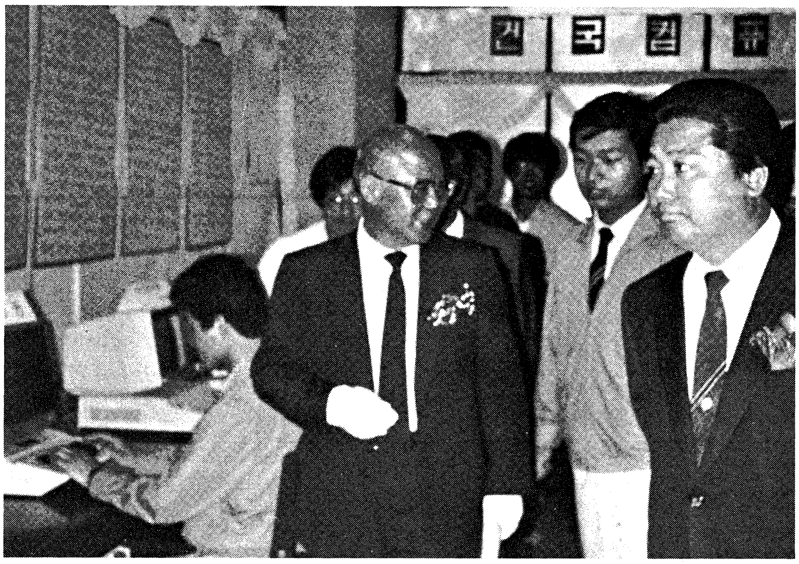
지난 1982년 전산과를 주축으로 설립된 건국대학교 컴퓨터연구회(KCA)는 현재 80여명의 회원으로 구성되어 있다.

동 서클에서는 회원간의 강한 유대관계를 가지고 개개인의 실력배양을 위해 프로그램 전시회를 매년 세번씩 실시하는 등 타 서클에 비해 매우 활발한 활동을 전개하고 있다.

제 14회 프로그램 전시회는 지난 88년 11월 9일부터 11일까지 교내 학생회관 2층에서 가졌다. 이번 전시회에서는 소프트웨어 부문으로 터보파스칼 버전 4.0에서 구현한 DM and 고객관리, 신문 구독자 관리 등의 업무용 프로그램과 보석글의 9핀 프린터 프로그램인 화일 프린터, MASM 버전 4.0에서 구현한 영문 패키지에서 소프트웨어 한글을 사용케 하는 Screen Switcher, 그래픽 패키지의 화면을 하드카피 가능케 해주는 램상주 하드카피 프로그램, 터보파스칼 버전 3.01에서 구현한 COPY II CP/M 등의 유틸리티 프로그램이 선보였고, 하드웨어로는 많은 호응을 얻은 음성 응답기, 디지탈 카운터가 선보였다.

박병용회장(전자2)은 앞으로 동계방학을 이용한 언어 교육에서 좀더 많은 회원이 참여하도록 내용면에서 보강하고 실질적인 교육을 실시하며 외부적인 행사를 주관할 수 있는 KCA가 되며 써클 내부의 단합을 위한 행사를 위해 노력할 것이라고 한다.

총장님께서도 대단한 컴퓨터광이신지 지대한 관심을 보이셨다.



대한 치과의사 컴퓨터 동호인회
90년도까지 BBS 구축 예정

지난 1987년 4월에 발족된 대한 치과의사 컴퓨터 동호인회는 국내 치과계에 컴퓨터 인식의 보급과 효율적인 전산화를 도모하려는 목적으로 창립되었다.

현재 회원이 150명에 이르고 있는 이들의 모임은 매월 정기학술집담회, 세미나, 학술 강연회 및 단기연수모임을 통하여 컴퓨터 이용에 관한 전반적인 교육과 회원중에서 초보자부터 숙달한 사람에게 이르기까지 단계별 교육을 실시하고 있다.

동 모임의 사업으로는 컴퓨터 프로그램의 발표와 상호교환, 컴퓨터에 대한 정보수집과 프로그램 공동개발추진, 회원간의 친목과 경조, 컴퓨터의 사용과 구입 안내 등이 있다.

그동안 동 모임은 기존에 통일되지 않은 진료비 명세서를 통일시켜 병원업무의 효율성을 기하도록 하였으며, 기존 의료계의 컴퓨터 이용이 의료보험 청구에만 국한되었던 것을 교정치료, 환자진료에 이용할 수 있는 진단분석 프로그램의 도입을 시도하였다.

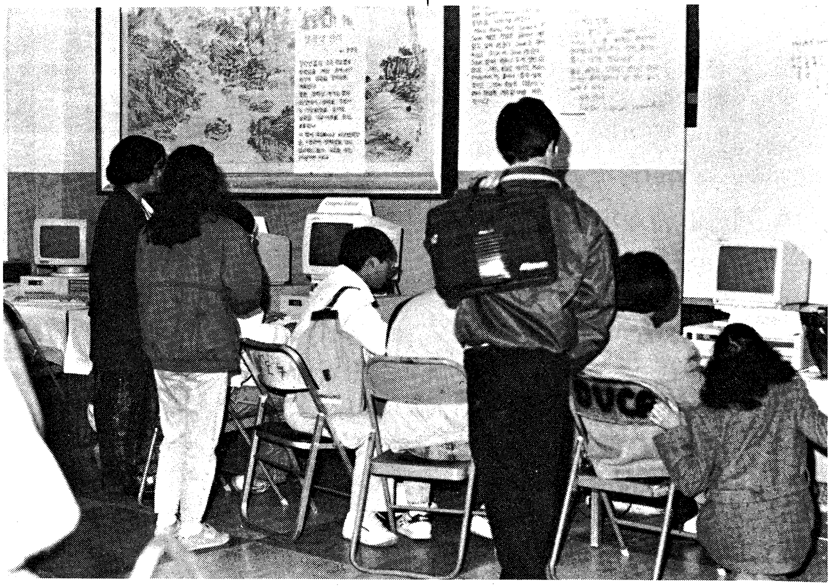
또한 데이콤의 전자메일에 가입하여 공중정보통신망과 연계하여 부산, 대전등 타지역에서 독자적으로 운영되고 있는 치과의사 컴퓨터 동호인회와 통신을 통해 정보를 교환하고 있다.

회원은 정회원과 명예회원, 준회원으로 구분되며 정회원은 대한민국 치과의사로, 명예회원은 당회의 발전에 공이 있는 사람으로, 준회원은 컴퓨터에 관심있는 사람으로 행사에 참여할 수 있다. 가입할 때 입회비는 3만원이며 특별한 정기회비는 없고 모임에서 일시적으로 수령하고 있다.

현재 임원진 구성은 회장(최종), 감사(김재권, 정상주), 고문(김규문), 부회장(홍종윤), 총무(박은기)와 8비트부의 이사 5명, 16비트부 이사 5명으로 되어있다.

앞으로의 계획은 바이오 리듬을 이용한 진료시스템을 도입할 것이며, 치과의사 BBS를 90년도까지 구축하여 회원간의 연락 및 공문발송등을 통한 친목도모는 물론 DNS를 통한 해외 치의학계 동향 및 최신 의료정보를 입수할 수 있도록 하며 빠른 시일내로 동모임을 학회 차원으로 확대하여 치과전산화를 합리적으로 도모함과 동시에 좋은 아이디어를 창출해 내기 위해 자체 사무실을 구비하여 활동인원 및 회원을 확대해 나갈 방침이라고 한다.

대한 치과의사 컴퓨터 동호인회에 대하여 자세한 사항을 알고자 한다면 박은기 총무(전화:(02)543-0219) 앞으로 연락하면 된다.



여학생들의 열렬한 참여의식으로 우먼파워를 발휘하였다.

동국대학교

제4회 프로그램 전시회 다양한 작품으로 높은 참여의식 표출

동국대학교 컴퓨터연구회(D.U.D.A)는 지난 11월 28일부터 12월 1일까지 교내 중앙도서관 1층에서 제4회 프로그램 전시회를 열었다.

이번 전시회에서는 터보C를 이용한 랩상주 프린터 제어, 풀다운 메뉴 유니트, 그래픽 에디터, 시스템의 주변장치와 메모리상태를 체크해 주는 도스 유틸리티, 지하철 이용에 사용되는 생활정보 프로그램과 터보 파스칼을 이용한 주소록관리 프로그램, 달력, 러브테스트, 오락용인 카드 맞추기, 넘어가기, Pick a Pare, 퍼즐게임과 베이직을 이용한 성적관리, 모뎀을 이용한 통신프로그램, 지방쓰기, 어셈블리 교육 프로그램과 디베이스와 클리퍼를 이용한 장학생 관리, 교회종합행정 등 다양한 작품이 선보여 회원들의 높은 참여의식을 표출하였다.

김광호회장(산공 2)은 다음의 전시회를 위하여 특강 및 단계별 그룹 스터디를 강화하여 질적인 향상을 가져오도록 하며 회원이 늘어나면서 소홀해지기 쉬운 선후배 관계를 돈독히 하는 일에 노력할 것이라고 한다.

DUCA는 지난 84년 5월에 창립된 이래 회원 상호간의 친목 및 유대를 강화하고 학술적인 활동을 위해 내적으로는 MT 및 체육대회, 야유회를 통하여 회원간의 친목을 도모하고 있으며 매주 5회의 스터디를 통하여 회원을 교육하고 있다.

또한 외부적으로는 써클의 홍보 및 실력향상을 위하여 전시회와 특강을 실시하고 있다.

특히 84년부터 시작되어 매년 동계와 하계 방학을 이용하여 실시하는 특강은 수강생이 교내로 국한되지 않고 30개교의 타교 수강생이 포함되어 200여명에 이르는 비교적 높은 호응도로 컴퓨터 대중화와 컴퓨터언어 교육이라는 써클의 목적을 달성시키면서 DUCA의 재정원이 되고 있다.

DUCA의 제9회 특강은 88년 12월 21일부터 89년 2월 17일까지 동국관 L 301강의실에서 오전 10시부터 12시 30분까지 실시하며 회비는 2만원이며 실습 사용 기종은 CYBER 930, 과목은 포트란 77, 코볼, 베이직 등이고 강사는 모든 컴퓨터 그래픽 시스템의 남궁 일주씨이다. 접수는 구내 매점에서 하고 있으며, 자세한 사항은 전화번호 (02)267-8131(교환 3427)로 문의하면 된다.

유니코사
23개교 4천여명이 모이는 연합체로 제2의 도약시도

엘빈 토플러는 그의 저서 '제3의 물결'에서 새로운 시대의 도래를 예언했고, 그의 예언은 정보화 시대로 이제 우리 눈앞에 실현되고 있으며 이 정보화 시대를 이끌어 갈 주역인 젊은 지성들이 유니코사라는 이름으로 모여 새로운 차원의 학생운동인 정보화운동을 전개해 나가고 있다.

유니코사는 1972년 고려대, 단국대, 서울대, 연세대, 홍익대 등 5개 대학이 유니코스(UNICOS:대학 전산학 연구회)라는 이름으로 '컴퓨터'란 단어가 사람들에게는 생소하게 들리던 시기에 컴퓨터란 미래의 학문을 연구하고자 창립되었다.

초기에 이들은 컴퓨터의 전반적인 인식이 자리잡지 못한 시기라 주로 개괄적이고 부분에 국한된 활동과 친목 위주의 활동을 해왔다.

그뒤 점차 시간이 지남에 따라 컴퓨터가 친밀하게 대중속으로 스며들자 유니코스는 적극적인 학술활동을 전개하기 시작했고 그러한 활동의 일환으로 1981년 유니코스는 유니코사로 개칭하면서 국제 사회의 시대적 요청에 부응하여 궁극적으로 국가 사회 발전에 기여한다는 전제아래 제2의 변신을 시도한다.

현재 23개 회원교의 회원 4천여명으로 연합된 유니코사는 회장과 총무, 회계, 서기, 기획부장, 섭외부장, 편집부장, 학술부장 등의 임원진으로 구성되어 있고, 총무와 운영위원회, 집행위원회, 학술위원회, 각부 부장회의 등의 기구를 두어 독자적이고, 효율적인 써클 운영을 하도록 조직되어 있다.

유니코사의 정기적인 모임은 각 학교의 대표격인 집행위원회, 학술위원회, 운영위원회의 모임으로 각 학교의 정회원으로 구성된 집행위원회가 매주 목요일마다 있고 이 모임에서는 운영위원회의 결정사항에 대한 실행과 당년도 활동의 사전 준비 및 사후 결산 보고와 회장 및 임원 선출, 그리고 기타의 발전을 위한 활동을 준비하고 있다.

학술위원회는 각 학교 학술담당 부장으로 구성되고 학술적인 일의 사전 준비계획 및 사후 보고를 위해 매달 2회의 모임을 갖고 있으며, 각 학교의 회장 및 책임자로 구성된 운영위원회는 당년도의 행사 일정과 집행위원회의 의견을 결정하는 일을 담당하기 위하여 매달 1회의 모임을 갖고 있다.

유니코사의 활동을 보면 학술활동으로 회원교간의 교류 및 친목도모를 위해 자체적으로 준비하는 매년 3차례의 학술 발표회가 있고, 유니코사 회원들의 실질적인 실력향상 및 대외 홍보와 정보교류를 위한 연합전시회가 있으며 사회의 저명인사 및 교수님을 초빙하여 정보화 사회에서 젊은이의 나아갈 방향이나, 컴퓨터에 관한 제반사항들을 청취하는 초청강연회가 있고 써클이 갖는 특성인 회원간의 내적인 결속을 위하여는 체육대회와 MT가 있다.

특히 86년부터 시작된 연합전시회는 업무용 프로그램, 교육용 프로그램, 유틸리티, 생활정보에 관한 프로그램, 오락 프로그램과 기타 프로그램등의 다양한 소프트웨어와 하드웨어가 전시되고 있고 매년 그 내용면과 질적인 면에서 향상되고 있으며 유니코사의 대부분의 회원들이 각 학교 컴퓨터 써클의 회원으로 구성되고 있으므로 전공을 하고 있는 회원보다는 비전공 회원이 주류이기 때문에 회원들에게는 매우 의미있는 행사로 유니코사의 저력을 과시하는 장이 되고 있다.

회원들의 교육은 EDPS, 베이직, C, 파스칼 등이 각 학교 단위로 행해지고 있고 전 회원을 대상으로 H-mail 사용법, MS-DOS 교육과 컴퓨터관련 업체의 견학 등 그때마다 필요한 사항에 대하여 실시하고 있고 신입생을 위하여 교육지를 발간하여 내실있는 교육을 하고 있다.

그리고 회원교 및 회원이 확대됨에 따라 전 학교의 평균적인 실력향상을 위해 그룹별 교육을 강화하려는 방침이라고 한다.

유니코사에 가입하려면 우선 유니코사의 목적에 부합되는 대학 컴퓨터 써클로서 학교의 승인을 받은 공식 써클이어야 하며, 2년이상의 활동실적이 있어야 하고, 정회원이 30명이상 등록되

꿈을 먹는 젊은이로 미래를 열겠다는 맑은 눈빛과 환한 웃음을 지닌 유니코사의 가족들



어 있어야 하며 신규가입에 관한 제반 서류를 구비하여 제출하게 되면 운영위원회에서 재적위원 3분의 2이상의 출석과 출석위원 3분의 2이상의 찬성으로 가결되며 한 대학에서 한 써클만이 가입하도록 제한하고 있다. 또한 회원에 가입하게 되면 회비(써클이 대표로 납부)를 매학기마다 2만원을 납부하여야 한다.

초창기에 비하여 이제는 자체 사무실을 구비하고 있어 실질적인 활동에 박차를 가할 수 있으며 데이타통신의 지원으로 각 학교마다 통신장비가 마련되어 있어 컴퓨터를 통한 정보교류를 행하고 있는 유니코사는 현재 주로 경인지역의 학교로 국한된 회원교를 전국적으로 확산시켜 전국적인 연합써클을 형성하려는 계획을 구상하고 있고 현재 부산지역, 대구지역, 충청지역, 전남지역 등의 컴퓨터 써클과 연합행사 주관을 위해 만남을 시도하고 있다.

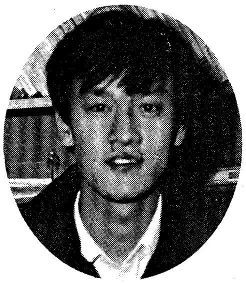
앞으로의 계획으로 유니코사는 자체의 BBS를 설치하여 보다 발전적인 정보통신을 시도하여 이 분야에 대한 실력을 배양하기 위해 주력할 것이며 외국과의 교류를 통하여 한국 젊은이들의 저력과시 및 정보교환을 시도할 것이라고 한다.

또한 유니코사는 자체활동의 단계를 벗어나 일반인들에게 컴퓨터인식을 심어주는 교육등을 실시하여 정보문화 확산을 위한 사회환원의 측면에서 활동할 것이라고 한다.

'꿈의 유니코사'라고 자신들을 지칭하면서, 젊기 때문에 무한한 미래에 도전할 수 있고, 그 도전을 실현하려는 의지가 결코 헛되지 않게 자신들의 꿈들을 하나씩 실현시킨 유니코사. 이들이 있으므로 해서 정보화시대를 살아가는 우리의 미래가 밝다는 것을 다시 한번 확신해 본다.

유니코사에 관한 자세한 문의 사항은 전화번호(02)791-1285로 연락하면 된다.

●미니인터뷰/배인식/88년도 유니코사 회장●



더 나은 정보화사회에 대한 동경과 그것을 실현하려는 의지가 필요합니다.

"유니코사가 커지면서 회원교간의 관심과 유대관계가 약해지는 것을 배제하고 회원간의 활발한 교류 및 유니코사에 대한 애착심을 고양하여 하나된 힘을 갖고자 노력한 한해였읍니다"라고 유니코사 회장으로 보낸 1년을 되돌아보고 아쉬운 감회에 젖는 배인식(국민대 금속과 3)씨.

실력있는 유니코사를 만들기 위해 종래의 행사에 변화를 주면서 88년 연합전시회를 철저하게 준비하였는데 올림픽관계로 대외적인 홍보에 욕심을 낼 수 없어 아쉬웠지만 전시된 여러작품의 높은 수준과 전시회때 어느 나이 많으신 분의 컴퓨터에 대한 열의로 감동을 받아 색다른 의미를 지닐 수 있었다고 한다.

그는 회장을 맡으면서 어려운 일도 많았지만 그때마다 격려를 아끼지 않았던 선배와 동기 그리고 후배들로 인해 용기를 얻었고 대학생으로 여러가지 경험과 많은 사람과의 만남을 통해 개인적으로는 새로운 자신감을 얻게 된, 인생에 있어서 가장 귀한 경험을 가질 수 있었다고 한다.

한가지 자랑스러운 점은 유니코사의 회원들이 올림픽 전산망의 자원봉사요원으로 활동하여 실질적인 개발은 아니지만 일부분의 역할을 수행했다는 것이라고 한다.

유니코사가 계속해서 발전하려면 회원간의 결속 및 유대관계를 돈독히하며 과거의 선배들이 동경하였던 생각들은 현재에 실현시킨 사실들을 토대로 더 발전된 정보화사회에 대한 동경과 그것을 실현하려는 의지가 있어야 하며 사회 각계에서는 새로운 각도의 관심을 가져야 한다고 강조한다.

91년부터는 현재와는 다른 차원의 행사가 진행될 것으로 예상하고 있으며 새로 선출된 임원들은 창조적이고 발전된 생각으로 동계방학동안 그 준비를 해야할 것이라고 덧붙인다.

"유니코사는 미래에 대한 선택입니다. 유니코사를 선택하였다면 자신의 선택을 개발하고 이용해야 하며 유니코사를 통해 자신과 더 나아가는 사회를 위해서 무엇인가 이루겠다는 신념을 지녀야 합니다"라고 힘주어 말하는 배인식씨의 유니코사에 대한 당부는 비단 유니코사 뿐만 아니라 정보화시대를 살고 있는 현대인에게 향한 당부가 아닐런지.

서울의대 컴퓨터 써클

의학도+컴퓨터 맹신자들의 집단

서울의대 컴퓨터 써클(C.O.M)은 단순히 컴퓨터에 대한 관심과 흥미로 모이기 시작해서 컴퓨터를 좀더 효율적으로 이용하고자 지난 88년 3월에 창립되었다.

COM은 서울의대에서는 최초의 컴퓨터써클로 맨처음 11명으로 시작하여 현재는 25명으로 구성되어 있다.

바쁜 학과시간의 틈을 내어 모이고 있는 이들은 매주 월요일마다 정기적인 모임을 가지고 있으며 시간의 제약을 받는 학기중에는 로터스, 데이터베이스, 사이드 킥 등의 응용프로그램의 사용법을 익히며 방학중에는 파스칼이나 데이터베이스 프로그램작성에 관한 세미나 및 교육을 실시하고 있다.

또한 신입생을 위하여는 컴퓨터를 사용함에 있어서 유저로서 사용하기 보다는 프로그래머로서 사용할 수 있도록 프로그램언어, 워드프로세서, 스프레드 시트, 데이터베이스 등을 1년의 교육과정으로 실시하고 있다.

앞으로의 계획으로는 잘 알려지지 않은 좋은 응용 프로그램을 편역할 예정이고 동계 방학을 이용하여 신입생 교육과정을 계속 진행해 나가며 인적자원을 확보하기 위해 컴퓨터 사용자들중에 의예과생들이 많다고 보고 경험자인 의예과생들도 모집할 예정이라고 한다. 또한 신입생 위주의 써클 운영에서 경험자들을 위한 교육방안을 모색할 것이라고 한다.

변성완회장(본과 2학년)은 자신들의 모임이 아직은 순수한 아마추어들로서 컴퓨터를 이용하고 있지만 좀더 깊은 지식을 습득하여 의학에 적합한 소프트웨어를 개발할 수 있도록 노력할 것이라고 한다. 문의는 서울의대 생리학교실 안철수조교((02)765-2210) 앞으로 하면 된다.

캐드 아카데미 유저그룹

여러가지 문제점을 공동으로 해결하고자 창립해

캐드 아카데미 유저 그룹은 전자, 기계, 건축등의 분야에 쓰이던 캐드가 범용성의 경향이 두드러지자 캐드를 사용할 때의 문제점과 노하우에 직면했을 때 경험자가 초보자에게 문제점을 해결할 수 있는 방안제시 및 사용하고 있는 분야에 적합한 캐드를 개발하는 것이 개인보다는 공동의 힘으로 개발하는 것이 바람직하다고 생각하는 사람들이 모여 지난 88년 9월 9일에 창립하였다.

아직까지는 개인보다는 업체별로 모이고 있으며 현재 회원으로 가입한 업체가 약20개 업체 정도에 이르고 있다.

매월 1회의 정기모임을 갖고 있고 회원의 교육은 현재는 초기의 단계로 캐드사용에 관한 토론방식으로 행해지고 있으나 앞으로는 세미나와 경험자의 노하우 개방에 관한 발표회 형식으로 진행할 것이라고 한다.

현재 임원진 구성은 회장(정윤용:한국자동기 연구소 대표), 부회장(김현득: 한양전설 대표), 간사(정완재: 삼환까뮤 PC부 대리), 총무(정양기:CADCOM SYSTEM 차장)로 되어 있고 회원에 가입하려면 캐드 캠 실사용자, 관련제품 제조 및 판매업체, 연구개발 단체, 정보도입 및 분석업체의 단체와 관련있는 학계와 업계의 단체 및 개인으로 매월 1만원의 회비를 모임이 있을 때마다 납부하면 된다.

이 모임의 사업으로는 캐드 캠의 선진기술의 도입, 도입된 선진기술의 개량연구개발 및 성과의 관리, 회원을 위한 캐드 캠 기술개발 실시 및 성과관리, 기타 관련기술의 연구개발 등이 있다.

또한 회원이 지방 및 다양한 분야로 확대될 것으로 예상하고 이에 따라 분야별, 구역별, 파트별 등 소그룹을 활성화시킬 계획이라고 한다.

캐드 아카데미 유저 그룹의 자세한 사항은 정양기 총무((02)583-3367) 앞으로 문의하면 된다.

-알려-드립니다-

본지에서는 써클 동정을 통해 전국에 산재되어 있는 아마추어들의 모임을 소개하여 동호인끼리 컴퓨터 이용자 모임을 형성할 수 있도록 유도하고 있읍니다.

이용자 모임을 알리고자 하는 분은 본지로 연락해 주시기 바랍니다.

연 락 처:
서울시 용산구 갈월동 17번지
세기빌딩 ㊄140-150
전화번호:(02)754-1044
FAX:(02)757-1049

순차화일과 랜덤화일

이기성/장왕교재연구원 상무

인터프리터 방식이란

사람이 입력한 프로그램을 컴퓨터가 해석하는데는 2가지 방식이 있다. 하나는 컴파일러 방식이며, 또 다른 하나는 인터프리터 방식이다. 컴퓨터가 처음 발명되었을 때는 컴퓨터 전문가(프로그래머)들만 프로그램을 작성할 수 있었다. 즉, 일반인이 알아보기 힘든 어셈블리어, 코볼, 포트란 같은 언어로 프로그램을 짰는데 이러한 언어들을 일반적으로 컴파일러형 언어라 한다. 컴파일러형 언어는 프로그램을 완전히 다 작성한 다음에 이 프로그램 전체를 한꺼번에 기계어로 바꾸어서 컴퓨터가 해석할 수 있도록 한다.

반면에 인터프리터형 언어는 컴퓨터 전문가가 아닌 일반인들도 쉽게 프로그램을 작성할 수 있도록 개발된 것이다. 프로그램을 완전히 다 작성한 뒤에 한꺼번에 기계어로 바꾸는 방식이 아니고, 프로그램을 한줄씩 작성하면 바로 한줄씩을 기계어로 바꾸어 컴퓨터가 이해할 수 있도록 하는 방식이 바로 인터프리터 언어의 특징이다. 즉, 100개의 행으로 구성된 프로그램이 있다면 행을 기계어로 바꾸어 컴퓨터를 실행시키고, 다음에 2번째행을 기계어로 바꾸어 컴퓨터를 실행시키고, 다음엔 3번째행을 기계어로 바꾸고…, 이런 식으로 100번을 한행씩 차례로 기계어로 바꾸어서 컴퓨터에게 명령을 실행시키는 것이다. 컴파일러 언어로 작성된 100행짜리 프로그램은 한번에 100줄 모두를 기계어로 바꾸어 컴퓨터에게 주게 된다.

컴파일러언어를 기계어로 바꾸려면 2개의 과정을 거친다. 먼저 기계어 코드화일(Object File)로 바꾸어 이것을 완전한 명령화일(Execution File)로 다시 한번 바꾸는 과정(Link)을 거치는 것이다.

그러나 인터프린터언어는 기계어로 바꾸는 과정이 간단하다. 그냥 한번의 과정으로 족한 것이다. 단지 인터프리터언어로 작성된 100행짜리 프로그램은 한번에 한행씩 기계어로 바꾸어 컴퓨터에 명령을 주는 작업을 100번 반복하므로 컴파일러언어에 비해 시간이 더 걸리는 단점이 있다(그림 1 참조).

인터프리터방식 언어의 대표적인 예는 베이직이다. 베이직에서는 한번에 한줄씩(한 행:line number) 기계어로 바꾸어 컴퓨터에게 명령을 하므로, 사용자는 자기가 작성한 행의 실행결과를 바로 알 수 있다. 만일 자기가 원하는 대로 실행 결과가 나오지 않으면 그 행을 고치면 된다. DBASE언어는 정식 컴퓨터언어는 아니고 간이 컴퓨터언어지만 인터프리터상태(마침표 프롬프트 상태)에서 직접 명령을 하나씩 줄 수 있으므로 인터프리터언어의 범주에 들어 간다.

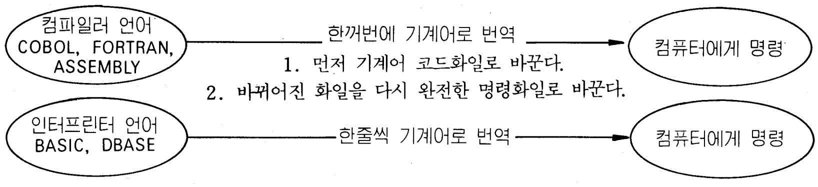
디스켓의 구조

컴퓨터의 본체에 달려있는 기억장치는 RAM이라는 손가락만한 덩어리로 이루어졌는데, 이것은 전기가 나가면 현재 기억하고 있던 내용을 모두 잃게 된다. 다시 말하면 지금껏 프로그램을 짜다가 전기스위치를 껐다가 다시 켜면 컴퓨터의 메모리는 텅 빈 상태로 되는 것이다. 길다란 프로그램을 작성하려면 며칠씩 작업하여야 하므로 오늘 작성한 프로그램의 내용을 전기가 꺼져도 지워지지 않는 장소에 보관시킬 필요성이 생긴다.

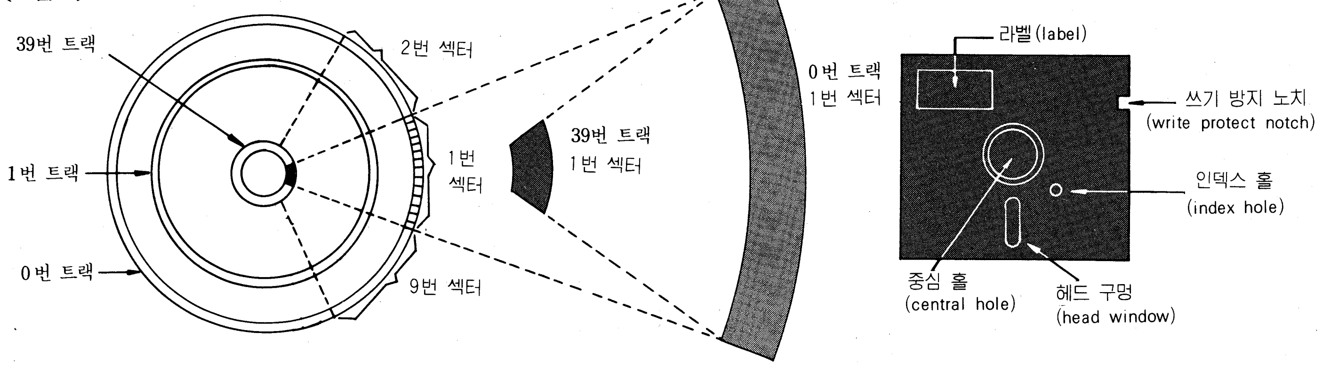
이러한 장치가 바로 디스크이다. 이것은 컴퓨터 본체를 주기억장치라 부르는데 반하여 보조기억장치라고 부른다. 보조기억장치에는 마그네틱 테이프, 플로피디스크, 하드디스크 등이 있는데 여기에서는 가장 일반인에게 많이 사용되는 플로피디스크를 주로 알아보기로 한다(그림 2 참조).

16비트 퍼스널컴퓨터에서 주로 사용

〔그림 1〕 컴파일러 언어와 인터프리터 언어



〔그림 2〕 5.25인치 플로피 디스크의 구조



되는 플로피디스크(이하 디스크라 한다.)는 5.25인치이다. 예전에 많이 사용되던 8인치 디스크와 1980년초에 개발된 3.5인치 디스크도 있지만 1976년에 슈가트(미국)에서 개발된 5.25인치 디스크가 우리나라에서는 가장 많이 사용된다.

5.25인치 디스크는 다시 기록용량에 따라 단면단밀도(1S), 단면배밀도(1D), 양면배밀도(2D,2DD)와 양면고밀도 (2HD)로 구분하는데, IBM-XT와 IBM-AT 에서는 양면배밀도를 주로 이용한다 (표 1 참조).

1섹터의 크기는 MS-DOS에서는 512바이트가 보통이다. IBM-XT용 디스크는 500K바이트라고 선전을 하나 실제로는 공디스크를 바로 사용하지 못하고 초기화(format)를 시키고 사용하여야 하므로 360K바이트만을 사용할 수 있다.

화일작성

화일에는 프로그램화일과 데이터화일이 있다. 데이터화일은 기록하는 방식에 따라 순차화일과 랜덤화일로 구분된다.(그림 3 참조)

• 순차화일 명령어

베이직 프로그램으로 만들고 읽을 수 있는 데이터화일에는 순차화일(Sequential File)과 랜덤화일(Random File)이 있다. 순차화일은 랜덤화일에 비해 화일을 만드는 방법이 간단하다. 반면에 많은 데이터가 기록된 순차화일에서 필요한 데이터를 찾을 때는 랜덤화일보다 속도가 느리다는 단점이 있다.

'SEQ-DATA'라는 순차화일을 만들고 읽는 보기를 **리스트** 1에 들었는데 20행에서 'SEQ-DATA'라는 화일을 열라고(open) 명령하고 50행에서 'PRINT' 명령으로 'SEQ-DATA'화일에다 '1, 100'에서 '5, 500'까지 기록시키고는 70행에서 화일을 닫는다(close).

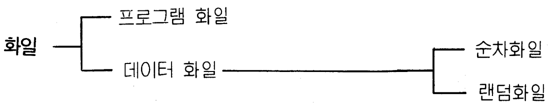
80행에서 데이터가 제대로 기록되었는지 확인하려고 'SEQ-DATA'화일을 연다. 100행에서 첫번째 데이터를 꺼내서(input) 110행으로 가서 화면에 보여주고(print) 120행에서 'GOTO 90' 명령을 만나 다시 데이터를 찾는다. 이렇게 5번을 반복하면 데이터가 더 이상 없으므로 순차화일의 끝 표시인 CHR$(26)을 만나게 된다. 이 끝 표시는 순차화일에서 컴퓨터가 자동으로 만들어 주는 것이다. 90행에서 화일의 끝 'EOF(1)'이면 화일을 닫고 프로그램을 끝낸다(그림 4 참조).

• 랜덤화일 명령어

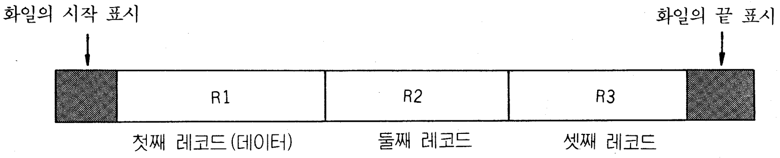
랜덤화일은 순차화일과는 달리 데이터를 찾을 때 처음부터 순서대로 찾는 것이 아니고 해당 레코드 번호를 바로 찾는 방식이므로 찾는 속도가 빠르고 실제 업무에서 많이 사용된다.

비록 화일을 만드는 단계가 순차화일에 비해 조금 복잡하지만 일단 화일을 만들어 놓으면 아주 사용하기에 편리하다. 또한 디스크에 데이터가 기록되는 형태가 2진 코드이므로 아스키문자로 기록되는 순차화일보다 적은 장

〔그림 3〕 화일의 종류



〔그림 4〕 데이터가 3개 기록된 순차화일



소를 차지한다. 다시 말하면 같은 용량의 디스켓에도 랜덤화일의 데이터를 더 많이 기록할 수 있다.

길이가 60바이트인 랜덤화일 'RANDOM-D' 를 만들고 읽을 수 있는 프로그램이 **리스트** 2이다. 20행에서 화일을 열고, 30행에서 60바이트를 NA$, A$, PHONE$, JOOSO$의 4개의 항목으로 구분한다. 40행에서 90행까지는 한 개의 데이터(레코드) 내용을 입력받는 루틴이고 100행, 120행, 130행은 입력된 내용을 왼쪽으로 맞추는(LSET) 곳이고 110행은 '금일 판매액'을 단정도(Single-Precision)의 숫자로 바꾸어 문자열인 A$에 기억시키는 곳이다. 140행에서 앞에서 입력된 4개 항목을 CODE$라는 레코드번호로 'RANDOM-D'화일에 기록한다. 입력번호가 '0'이면 200행으로 가서 화일을 닫고(close) 화일을 기록하는 작업을 끝낸다(**그림 5** 참조).

300행에서 410행까지는 앞에서 기록한 화일내용을 찾아보는 루틴이다. 320행에서 60바이트짜리 'RANDOM-D' 라는 화일을 열고 330행에서 한 개의 레코드를 4개의 항목(Field)으로 나눈다. 340행에서 찾고자 하는 레코드번호를 입력받아서 360행으로 해당레코드를 읽는다(GET). 370행에서 400행까지는 찾은 데이터를 항목별로 화면에 보여준다. 참고로 **표 2**에는 순차화일 명령어를, **표 3**에는 랜덤화일 명령어를 보였다.

리스트 3과 같이 앞에서 살펴본 화일 명령어를 이용하여 100개의 레코드를 관리하는 간단한 재고 계산 프로그램을 만들어 보자. 랜덤화일 만들기와 읽기는 앞의 보기대로 하면 되나, 화일의 초기화 루틴은 새로운 것이므로 간단히 살펴보기로 한다.

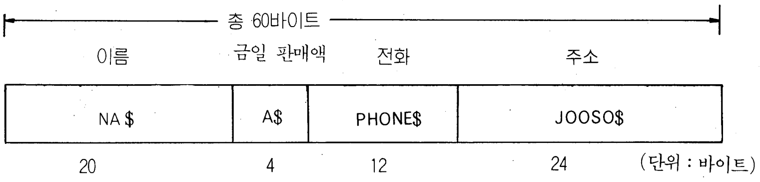
300행에서 LKS$에 INPUT$(1)로 한 바이트만 받아서 받은 내용을 VAL(LKS$)로 숫자 형태로 바꾼다. 320행에서 화일의 초기화를 선택하였으므로 FUNCTION 변수에는 '1'이 들어 있다. 1번에 해당하는 950번행으로 갔다 온다.('ON 1 GOSUB 950).

950행에서 데이터를 처음 입력하는가를 확인한다. 왜냐하면 이 루틴은 새로 화일을 만드는 곳이므로 만일 같은 이름의 데이터화일이 이미 있을 경우에는 기존 데이터를 모두 지우고 새로 만들게 되기 때문이다. B$에 대문자 Y를 넣으면 1000번 행으로 가서 화일을 만들기 시작한다.

〔표 1〕 양면 배밀도 디스크의 용량 (1섹터=512바이트)

섹터의 수	트랙의 수	면수	계산용량(바이트)
9	80	2	512×9×80×2=360K
16	160	2	512×16×160×2=1,250K

〔그림 5〕 RANDOM_D 화일의 한 레코드(데이터)의 구조



〔표 2〕 순차화일 명령어

IBM 컴퓨터	설 명
필요 없음	도스 명령어 표시
OPEN "SEQ DATA"	순차화일 열기
FOR OUTPUT AS #1	쓰기 전용으로
PRINT #1, 10	10이란 내용을 기록함
OPEN "I", #1, "SEQ_DATA"	순차화일을 읽기 전용으로 열기
INPUT #1, A	한개의 레코드를 읽음
PRINT A	화면에 표시
CLOSE	화일 닫기

〔표 3〕 랜덤화일 명령어

IBM 컴퓨터	설 명
필요 없음	도스 명령어 표시
OPEN "RANDOM_D"	랜덤화일 열기
AS #1 LEN=60	길이 60바이트
FIELD #1, 20 AS NA$, 4 AS A$, 12 AS PHONE$, 24 AS JOOSO$	60바이트를 4개로 구분한다
LSET NA$=NA1$	문자열을 왼쪽으로 정렬시킴
LSET A$=MKS$(WON)	단정도 숫자로 바꾸어 줌
PUT #1, CODE%	CODE%번째 레코드에 기록하기
GET #1, CODE%	CODE%번 레코드를 읽기
PRINT NA$	화면에 표시
CVS(A$)	단정도 숫자를 문자열로 바꿈
CLOSE	화일 닫기

〔리스트 1〕 순차화일 만들기와 읽기

```
10 REM SAVE"SEQ",A
20 OPEN "SEQ_DATA" FOR OUTPUT AS
   #1
30 FOR I = 1 TO 5
40      LK = I * 100
50      PRINT #1,I,LK
60 NEXT
70 CLOSE
80 OPEN "I",#1, "SEQ_DATA"
90      IF EOF(1) THEN GOTO 200
100     INPUT #1,A,B
110     PRINT A,B
120     GOTO 90
200 CLOSE :END
```

이미 60행에서 화일을 열었기 때문에 1000행에서 CHR$(255)를 F$에 넣어 준다. CHR$(255)는 공백(아무 내용이 없는 빈 것)이므로 나중에 프로그램에서 해당 레코드가 빈 것인지 아니면 제품이 이미 등록이 되어 있는지를 구별할 때 사용한다(그림 6 참조).

1010행에서 1040행까지는 LJAIGO.DAT화일에 100개의 빈 레코드를 만들어(PUT #1,1 -- 100) 준다. 1070행에서 RETURN을 만났으므로 GOSUB 950으로 왔던 320행의 다음을 실행하러 330행으로 간다. 여기서 FUNCTION은 현재 '1'이므로 340행으로 내려가서 화면에"아무 키나 누르면 차림표로 돌아갑니다."를 표시한다. 아무 키나 눌러주면 키값을 INKEY$가 받아서 AA$에 넣어 주고는 GOTO문에 의해 80행으로 분기하여 CLS명령으로 화면을 지우고 차림표를 화면에 다시 그린다.

모니터와 프린터제어명령

베이직 명령어에서 가장 기초적인 것 중의 하나가 PRINT명령이다. 이 PRINT명령은 모니터화면에 보이는 것, 프린터용지에 인쇄하는 것, 디스크에 기록하기 위하여 순차화일용 버퍼에 보관하는 것의 3가지로 사용된다 (표 4 참조).

• 모니터 제어 명령

앞에서 살펴본 PRINT명령어는 출력용 명령어이다. 모니터에 출력시키려면, 즉 모니터 화면에서 결과를 보려면"PRINT"를 사용하고 프린터용지에 인쇄하려면"LPRINT"를 사용하면 되었다. 이것과는 반대로 입력용 명령어가"INPUT"이다. INPUT명령어는 컴퓨터의 키보드로부터 데이터를 받아들인다. 프로그램 중에서 INPUT명령어가 나오면 모니터 화면에 물음표(?)가 나타나면서 키보드에서의 입력을 기다린다. 다만 INPUT 바로 뒤에 따옴표가 나오고 안내문이 있으면 그 안내문을 모니터 화면에 보여주고 나서 물음표가 나타난다. 만일 물음표가 보기 싫으면 안내문 뒤의 따옴표 뒤에 세미콜론(;)대신 쉼표(,)를 붙이면 된다.

이번에는 INPUT으로 데이터를 받아들여서 PRINT로 화면에 나타내는 방법을 알아보자. 숫자 데이터의 경우는 PRINT USING 명령어를 사용하면 소수점자리도 맞출 수 있다. 프린트를 하려면 LPRINT USING을 사용하면 된다.

이밖에도 모니터 화면을 제어하는 명령어에는 LOCATE와 COLOR명령어가 있다. 화면의 위에서 몇번째 줄, 앞에서(왼쪽에서) 몇번째 칸에다 표시하려는가에 따라서"LOCATE 줄 수, 칸 수"로 명령을 준다. COLOR명령은

〔리스트 2〕 랜덤화일 만들기와 읽기

```
10 REM SAVE"RANDOM",a
20 OPEN "RANDOM_D" AS #1 LEN = 60
30 FIELD #1,20 AS NA$,4 AS A$,12 AS PHONE$,24 AS JOOSO$
40      INPUT "입력 번호    ( 1 - 99 , 0= 끝 ) : ",CODE%
50      IF CODE%=0 THEN GOTO 200
60      INPUT "이 름            : ",NA1$
70      INPUT "금일 판매액      : ",WON
80      INPUT "전화 번호 ☎      : ",PH1$
90      INPUT "주 소            : ",JOO1$
100     LSET NA$ = NA1$
110     LSET A$ = MKS$(WON)
120     LSET PHONE$ = PH1$
130     LSET JOOSO$ = JOO1$
140 PUT #1, CODE% :REM 화일에 레코드를 기록하기
150 GOTO 40
200 CLOSE
300 REM 랜덤 화일 읽기
310 REM
320 OPEN "RANDOM_D" AS #1 LEN = 60
330 FIELD #1,20 AS NA$,4 AS A$,12 AS PHONE$,24 AS JOOSO$
340      INPUT "찾는 번호    ( 1 - 99 , 0=끝 ) : ",CODE%
350      IF CODE% = 0 THEN GOTO 500
360      GET #1, CODE% :REM 해당 레코드 읽기
370      PRINT NA$
380      PRINT USING "###,###,###";CVS(A$)
390      PRINT PHONE$
400      PRINT JOOSO$
410 GOTO 340
500 CLOSE: END
```

〔그림 6〕 LJAIGO.DAT 화일의 레코드 구조

총 41바이트

	품목 이름	재고수량	최저 재고수	단가
F$	LD$	Q$	R$	P$
1	30	4	2	4

(단위 : 바이트)

〔표 4〕 PRINT 명령어의 용도

출력 장소	명 령 어
모니터 화면	PRINT "장왕교재 연구원"
프린트 용지	LPRINT "장왕교재 연구원"
순차화일	PRINT #1, "장왕교재 연구원"
모니터 화면	PRINT LK$
프린트 용지	LPRINT LK$
순차화일	PRINT #1, LK$

자기가 사용하는 모니터와 모니터를 콘트롤하는 카드(컴퓨터 내부에 꽂혀 있음)의 종류에 따라 다르다. 허클리스 카드가 꽂힌 컴퓨터의 그린모니터에서는 정상 글자를"COLOR 7, 0"로, 역상 글자는"COLOR 0, 7"로 명령을 준다. 앞의 랜덤화일 프로그램의 리스트를 보면 80행이"CLS" 명령으로 화면을 지우는 것이고, 90행이"LOCATE 3"으로 위에서부터 3번째줄로 커서를 내려오는 것이다. 90행 뒷부분의"COLOR 0,7"은 역상화면으로 바꾼 것이다. 110행이 역상화면에다 앞에서 20칸을 띄고"재고 관리 차림표"라고 쓰는 것이고 120행이"COLOR 7,0"로 다시 정상화면으로 돌려주는 것이다.

● 프린터 제어 명령

프린터는 인쇄속도나 인쇄하는 방식 등 여러 종류로 구별되지만 우리가 취급하는 퍼스널컴퓨터에 주로 사용되는 프린터는 도트매트릭스 방식의 9핀 프린터와 24핀 프린터이다. 물론 레이저 프린터나 열전도 방식의 프린터도 사용은 되고 있으나 아직까지 퍼스널컴퓨터에서는 도트 매트릭스 방식이 많이 사용된다. 도트 매트릭스 방식의 프린터는 우리나라에서 엡슨 프린터 기종과 큐닉스(후지쯔) 프린터 기종이 많이 사용되므로 이 2종류에 대하여 살펴보기로 하자.

프린터에서 한글을 사용할 경우에는 프린터 내부의 스위치를 한글 상태로 맞추어 놓은 것과 프린터 자체는 영문자 사용 상태로 두었다가 한글을 사용할 경우에만 한글 상태로 조정하는 방법이 있다. 여기서는 프린터에 전원을 새로 넣었을 때는 영어 전용상태이다가 한글이 필요할 때만 프로그램에서 한글 상태로 만드는 경우를 알아본다. (표 5 참조)

프린터의 명령어는 프린터를 만든 회사마다 다른 것이 아니고 거의 비슷하므로 표 5에 나와 있지 않은 프린터를 갖고 있는 독자는 프린터 매뉴얼을 보고 명령어를 하나씩 실험해 보면 되니까 걱정할 필요가 없다.

같은 프린터라도 그 프린터의 롬(ROM) 에 어떠한 문자가 기억되어 있는가에 따라 모니터화면에서는 제대로 나오던 기호가 프린터로 인쇄하면 안나오거나, 다른 기호로 인쇄되거나 한다.

참고로 리스트 4에 엡슨프린터를 사용하여 출력을 제어하는 프로그램을 예로 보였다.

〔리스트 3〕 랜덤화일을 이용한 재고 계산 프로그램 일부

```
10 REM
30 CLS
60 OPEN "R",#1,"A:LJAIGO.DAT",41
70 FIELD #1,1 AS F$,30 AS LD$,4 AS Q$,2 AS R$,4 AS P$
80 CLS:REM ***** 차림표 시작 *****
90 LOCATE 3:COLOR 0,7
100 PRINT "                    "
110 LOCATE 3: PRINT TAB(20);"재고 관리 차림표"
120 COLOR 7,0
130 LOCATE 7,2
140 PRINT 1;"  화일의 초기화."
150 LOCATE 9,2
160 PRINT 2; " 새로 입력."
170 LOCATE 11,2
180 PRINT 3;"  화면에서 찾아보기."
190 LOCATE 13,2
200 PRINT 4;"  제품 입고."
210 LOCATE 7,42
260 PRINT 8; "끝. "
262 LOCATE 14,1
270 COLOR 15,0
280 LOCATE 17,30:PRINT "어느 번호 (1--8) : "
290 COLOR 7,0
300 LOCATE 17,52:LKS$ = INPUT$(1):FUNCTION = VAL(LKS$)
310 IF FUNCTION = 0 OR FUNCTION>8 THEN PLAY "CDE" :GOTO 80
320 ON FUNCTION GOSUB 950,370,520,620, 80, 80, 80,1080
330 IF FUNCTION = 8 THEN END
340 PRINT: PRINT".......아무 키나 누르면 차림표로 돌아갑니다. "
```

```
350 AA$ = INKEY$:IF AA$ = "" THEN 350
360 GOTO 80
370 ' 신규 입력
380 CLS
390 GOSUB 920
400 IF ASC(F$)<>255 THEN INPUT"이미 있읍니다. 지우고 그 자리에 다시
기록하시겠읍니까 <Y/N> :",A$:IF A$ <> "Y" THEN RETURN
410 LSET F$ = CHR$(0)
420 INPUT"품 목          이 름              : ",DESC$
430 LSET LD$ = DESC$
440 INPUT "재 고          수 량              ",Q
450 LSET Q$ = MKS$(Q)
460 INPUT"최저 재고수(30,000개 이하)",R%
470 LSET R$ = MKI$(R%)
480 INPUT "단          가                ",P
490 LSET P$ = MKS$(P)
500 PUT #1,PART%
510 RETURN
520 REM 화면에 표시
530 CLS
540 GOSUB 920
550 IF ASC(F$) = 255 THEN PRINT "이 품목은 기록되어 있지 않습니다.
    ":PLAY "CGE":RETURN
560 PRINT USING"품 목 번 호          ###";PART%
562 PRINT "----------------------------------------------
----------"
570 PRINT D$
580 PRINT USING "현 재고량    #,###,###";CVS(Q$);
582 PRINT "  개"
590 PRINT USING"최저 재고수      ##,###";CVI(R$)
600 PRINT USING"단 가       #,###,###";CVS(P$);
602 PRINT "  원"
610 RETURN
620 REM ADD-RTN
630 CLS
640 GOSUB 920:REM
650 IF ASC(F$) = 255 THEN PRINT " 이 품목은 기록되어 있지 않습니다.
새로 등록을 하세요.":PLAY "CFG":RETURN
660 PRINT D$;CVS(Q$):INPUT"추가 입고량을 기입하세요.",A
670 Q = CVS(Q$)+A
672 PRINT"현 재고량은 ";Q;"  개가 됩니다."
680 LSET Q$ = MKS$(Q)
690 PUT #1,PART%
700 RETURN
920 PRINT:PRINT:INPUT"***    품목 번호를 쓰세요  (1-100)  :
  ",PART%
922 PRINT "====================================================
======"
930 IF PART%>100 THEN PRINT "품목 번호는 100번이 최대입니다. ":
    BEEP:GOTO 920 ELSE GET #1,PART%:D$ = LD$ :RETURN
940 END
950 REM INIT-DATA-FILE
960 CLS:LOCATE 10,20:PRINT"데이터를 처음 입력하십니까 ?"
970 LOCATE 12,25:PRINT"Y = 데이터화일 새로 만들기. "
980 LOCATE 13,25:PRINT"N = 다시 차림표로. "
990 LOCATE 10,50:INPUT".",B$:IF B$<> "Y" THEN CLS:RETURN
 :REM 대문자 Y
1000 LSET F$ = CHR$(255)
1010 FOR I = 1 TO 100
1020 LOCATE 5,70:PRINT I
1030 PUT #1,I
1040 NEXT I
1050 LOCATE 20,10:PRINT "화일을 다 만들었읍니다."
1060 FOR PA = 1 TO 500:NEXT PA
1070 RETURN
1080 REM
1090 PRINT:COLOR 1,0
1100 PRINT TAB(5);"  감사합니다. "
1110 COLOR 7,0
1120 PRINT:PRINT
1130 RETURN
2000 END
2010 REM:EOF H2-JAIGO.BAS
```

〔그림 7〕 INPUT 명령어의 사용

```
10 INPUT LKS
20 INPUT "숫자를 넣으세요  ";LKS
30 INPUT "숫자를 넣으세요  ",LKS
40 END
OK

RUN
? 100
숫자를 넣으세요 ? 100
숫자를 넣으세요    100
OK
```

〔그림 8〕 INPUT 명령어와 PRINT 명령어

```
10 REM  test-77
20 A = 10
30 INPUT "숫자를 넣으세요
  ";LKS
40 PRINT A
50 PRINT LKS
60 PRINT A,LKS
70 PRINT A;LKS
80 PRINT "----+---10"
90 PRINT "A= ";A
100 PRINT "LKS= ";LKS
110 END
OK

run
숫자를 넣으세요 ? 4321
 10
 4321
 10
 10            4321
 10  4321
 ----+---10
A = 10
LKS=  4321
OK
```

〔그림 9〕 PRINT USING 명령어

```
10 REM
20 A = 10
30 B = 20
40 PRINT A
50 PRINT B
60 PRINT USING "  #####";A,B
70 PRINT USING "  ###.###";A,B
80 PRINT USING "  $##.##";A,B
90 END
OK

run
 10
 20
    10        20
 10.000      20.000
 $10.00      $20.00
OK
```

〔표 5〕 프린터별 명령어 요약

내 용	큐닉스(PC-24)	엡슨(삼보 RX, LQ-1500)
한글 시작	CHR$(27)+"h"+CHR$(1)	CHR$(27)+"h"
한글/영자 2 : 1 맞춤	CHR$(27)+"g"+CHR$(1)	CHR$(27)+"W"
한글/영자 2 : 1 맞춤 끝	CHR$(27)+"g"+CHR$(0)	CHR$(27)+"w"
가로로 2배 확대	CHR$(27)+"W"+CHR$(1)	CHR$(27)+"D", 또는 "L"
가로로 2배 확대 끝	CHR$(27)+"W"+CHR$(0)	CHR$(27)+"d", 또는 "N"
길게 2배 첫 줄	CHR$(27)+"i"+CHR$(1)	CHR$(27)+"〔"(LQ-1500)
길게 2배 두째 줄	CHR$(27)+"i"+CHR$(2)	CHR$(27)+"〕"(LQ-1500)
세로로 2배 길게 끝	CHR$(27)+"i"+CHR$(0)	—
역상(글씨가 희게)	CHR$(27)+"r"+CHR$(1)	CHR$(27)+"R"
역상 인쇄 끝	CHR$(27)+"r"+CHR$(0)	CHR$(27)+"r"
두꺼운 글자	CHR$(27)+"E"	
두꺼운 글자 끝	CHR$(27)+"F"	
밑줄친 글자	CHR$(27)+"-"+CHR$(1)	CHR$(27)+"U"
밑줄친 글자 끝	CHR$(27)+"-"+CHR$(0)	CHR$(27)+"u"
고딕체 한글	CHR$(27)+"m"+CHR$(1)	
고딕체 한글 끝	CHR$(27)+"m"+CHR$(0)	
축소 한글	CHR$(15)	CHR$(15)+
축소 한글 끝	CHR$(18)	CHR$(18)+
세로쓰기로 인쇄	—	CHR$(27)+"j"(LQ-1500)
프린터 띄어쓰기	CHR$(27)+"t007"	CHR$(27)+"T007"

〔리스트 4〕 엡슨 프린터(LQ-1500)의 테스트 프로그램

```
10 REM save"LQ-test",a
20 CLS
30 PRINT "프린터를 켜세요 "
40 PLAY"c"
42 LPRINT"PRINTER TEST"
43 LPRINT"ENG-MODE임. 한글 세트 안함."
50 LPRINT CHR$(27)+"h"
60 LPRINT"가로 2배 = CHR$(27)+'D' "
80 LPRINT CHR$(27)+"D" ;"가로 2배 이 강원";CHR$(27)+"d"
90 LPRINT"세로 2배 = CHR$(27)+'[','']' "
100 PRINT
110 LPRINT CHR$(27);"[";"세로 2배 이 영미"
120 LPRINT CHR$(27);"]" "세로 2배 이 영미"
130 PRINT"역상 인쇄 "
150 COLOR 0,7:PRINT "인버스  박 정자";:COLOR 7,0:
    PRINT"이제부터는 정상 글자"
160 LPRINT"역상 인쇄 = CHR$(27)+'R' "
170 LPRINT CHR$(27);"R"; "인버스   김 명희";CHR$(27)+"r"
180 PRINT"두껍게"
190 COLOR 15,0:PRINT "두껍게   진 진순";:COLOR 7,0:PRINT"
  (보통으로 환원)"
220 PRINT"밑줄 침"
230 COLOR 1,0:PRINT "밑줄 침. 정 진순";:COLOR 7,0:PRINT "밑줄 안침."
240 LPRINT CHR$(27);"U"; "밑줄 침.   이 연숙";CHR$(27)+"u";
      "이제부터는 밑줄 안침."
250 LPRINT
260 LPRINT "----+----1----+----2----+----3----+----4----+---
 -5----+----6----+----7----+----8"
270 LPRINT "한글 테스트입니다."
280 LPRINT CHR$(15);"축소 테스트입니다."
302 LPRINT CHR$(27);"h";
310 LPRINT "한글 테스트입니다. 다시 한글 세트했음."
400 LPRINT "세로로 문자 인쇄하기"
410 LPRINT CHR$(27);"J";"한글 테스트입니다. 지금은 세로로 인쇄중입니다.
  = CHR$(27)+'j' "
420 LPRINT CHR$(10): END
```

화일작성

박성현/공영종합시스템 개발부

디베이스의 세계에 들어온 독자를 위하여 디베이스의 역사와 더불어 디베이스의 몇가지 기능을 지난호에서 설명했다.

이번호에는 디베이스에서는 데이터를 어떤식으로 저장하고 어떻게 다시 찾아 볼 수 있는가를 공부해 보기로 하자.

데이터베이스 화일의 작성

데이터베이스 화일을 처음 만들기 위해서 다음의 명령을 입력해 보자.

```
CREATE <file-name>
```

그러면 데이터베이스의 새로운 구조를 만들기 위한 화면이 나온다. 여기서의 작동방법은 굳이 설명을 하지 않아도 키보드를 몇번 사용해 보면 쉽게 알 수 있을 것이다. 다음 화면 1은 데이터베이스 화일작성 화면을 나타낸다.

화일의 확장명을 지정하지 않았다면 자동으로 .dbf라는 화일명으로 붙는다. 필드명, 타입, 폭, 소수점 이하의 자리를 입력하므로써 1개의 필드를 지정할 수 있다.

필드의 타입은 첫글자(C,N,L,D,M)를 치거나 스페이스바로 고르고 리턴키로 선택하면 된다. 필드의 타입은 지난호에 설명한 변수의 타입과 거의 유사하나 필드의 타입에는 메모필드라는 것이 한가지 추가된다. 이는 특별한 용도에서만 사용하므로 이에 대한 설명은 다음으로 미루고 우선 데이터베이스 화일의 작성과 사용을 설명하고자 한다.

소수점이하의 자리는 수치필드에서만 들어간다. 소수점도 한바이트를 차지하므로 자리수 계산에 주의해야 한다.

2바이트

999.99

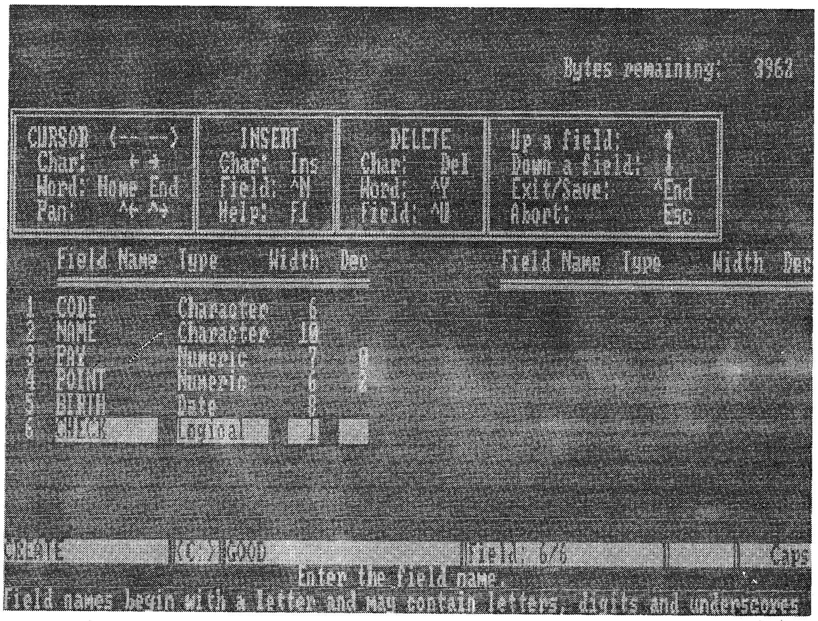
6바이트

위 포인트의 경우 전체 6바이트중 2바이트를 소수점 이하로 사용하고 1바이트는 소수점으로 나머지 3바이트만이 소수점 윗부분에서 사용된다.

위와 같이 입력한 후에 Ctrl-End를 누르거나 새로운 필드를 입력할 곳에서 리턴키를 누름으로써 입력을 끝낼 수 있다. 현재 데이터베이스 화일의 구조를 알아보기 위해서는 다음과 같은 명령으로 알아볼 수 있다.

```
. list structure
Structure for database: C:good.dbf
Number of data records:       0
Date of last update   : 11/01/88
Field  Field Name  Type       Width    Dec
    1  CODE        Character      6
    2  NAME        Character     10
    3  PAY         Numeric        7      0
    4  POINT       Numeric        6      3
    5  DATE        Date           8
    6  CHECK       Logical        1
** Total **                      39
```

〔화면 1〕 데이터베이스 화일작성 화면



이때 합계 바이트 수는 실제로는 보이지 않지만 데이터의 삭제표시를 위해 1바이트가 사용되므로 38보다 1이 더 큰 39가 된다.

데이터베이스 화일구조 변경

데이터베이스 화일의 구조를 변경하기 위해서는 다음과 같은 명령을 사용하며 그 사용방법은 데이터베이스 화일을 만드는 작업과 동일하다.

```
. modify structure
```

이때 현재의 데이터베이스 화일에 데이터가 들어 있을 경우 필드명과 필드 타입을 바꾸면 데이터를 잃어버리고 폭을 바꿀경우 폭을 줄였을 경우만 줄인만큼 잃어 버리게 될 것이다. 이젠 실제 데이터를 입력해 보자.

데이터의 입력

데이터를 입력하기 위한 명령으로는

```
. append
```

라는 명령을 사용한다. append라는 명령을 사용하면 화면에 필드의 크기만큼 반전되어 있는 부분에 커서가 위치하며 그곳에 데이터를 입력하면 된다. 우선 임의의 데이터를 5개만 입력해 보길 바란다.

이때 날짜를 입력하는 순서는 월/일/년의 순으로 입력해야 하며 논리필드인 CHECK는 T/F 또는 Y/N으로 입력이 가능하다.

입력을 중지하고자 할때는 새로운 레코드를 입력하기 위한 화면이 나왔을 때 어떤 데이터도 넣지 않고 리턴키를 치면 빠져 나오게 된다.

데이터베이스 화일의 사용

데이터베이스 화일을 즉시 만들었다면 방금 만든 화일이 열려 있어 이를 사용하게 되지만 새로이 디베이스를 구동시켰거나 데이터베이스 화일을 새로 사용하고자 할때는 새로이 사용하고자 하는 화일을 열어 두어야 할 것이다. good.dbf를 사용하기 위해 열어놓는 명령은

```
. use good
```

이고 이 명령으로 good.dbf라는 데이터베이스 화일을 열게 된다. 이때 디베이스는 열려 있는 모든 데이터베이스 화일을 닫고 good.dbf라는 데이터베이스 화일만을 열게 된다. 그러므로

```
. use
```

라는 명령만을 사용하면 데이터베이스 화일을 모두 닫아버리는 명령이 된다.

데이터 읽기

지금까지 넣었던 데이터를 다시한번 보기위해 다음의 명령을 사용하면 모든 데이터를 볼 수 있다(화면 2 참조).

다음엔 날짜의 형태를 우리가 사용하는 년/월/일 순으로 그 형태를 바꾸기 위해 다음의 명령을 입력해 보자.(화면 3 참조).

날짜의 형태가 바뀌었을 것이다. set date ansi라는 명령을 사용하므로써 이제부터는 날짜의 형태가 항상 년/월/일 순으로 나타나게 되며 입력할 때에도 년/월/일 순으로 입력할 수가 있다. 그러면 데이터를 5개만 더 입력해 보자.

```
. append
```

5개의 데이터를 입력한 다음에는 다음명령을 입력하고 다시 데이터를 5개 입력해 보길 바란다.

```
. set carry on
. append
```

set carry on상태에서는 마지막의 데이터가 새로운 데이터를 입력할 부분에 복사되어 나오게 되므로 연속해서 비슷한 데이터를 입력하고자 할 때는 이 상태에서 데이터를 입력하면 더욱 간편히 데이터를 입력할 수 있게 된다.

아무 수정도 하지 않고 첫번째 필드

〔화면 2〕

```
. list
Record#  CODE   NAME         PAY  POINT BIRTH    CHECK
      1  000001 윤 택 중   810000  89.00 10/10/37 .T.
      2  000002 이 홍 룡   400000  65.20 07/08/61 .F.
      3  000003 박 준 영   340000  45.00 08/06/63 .T.
      4  000004 채 수 태   600000  87.34 12/27/58 .T.
      5  000005 김 규 봉   390000  77.00 12/18/62 .F.^
```

〔화면 3〕

```
.set date ansi
.list
Record#  CODE   NAME         PAY  POINT BIRTH    CHECK
      1  000001 윤 택 중   810000  89.00 37.10.10 .T.
      2  000002 이 홍 룡   400000  65.20 61.07.08 .F.
      3  000003 박 준 영   340000  45.00 63.08.06 .T.
      4  000004 채 수 태   600000  87.34 58.12.27 .T.
      5  000005 김 규 봉   390000  77.00 62.12.18 .F.^
```

의 첫머리에서 리턴을 치면 빠져나오게 된다.

이 상태를 중지하기 위해서는 다음과 같이 입력하면 된다.

```
.set carry off
```

이제까지 15개의 데이터를 입력했다. 그러면 현재의 입력해 놓은 데이터를 찾아 보도록 하자 (화면 4 참고).

4번째 레코드를 찾기 위해서는 다음과 같이 입력한다 (화면 5 참고).

6번째 레코드를 가기 위해 다음과 같이 입력할 수도 있다 (화면 6 참고).

숫자만의 입력으로도 가능하다. 2번째 레코드로 가보자 (화면 7 참고).

코드가 [000012]보다 큰 데이터만을 찾아보자 (화면 8 참고).

이름이 '최'로 시작하는 사람만 찾아보면 (화면 9 참고).

이름이 가나다 순의 '전'씨보다 뒤편의 사람만을 찾아보면(전씨 포함해서) (화면 10 참고).

급여가 50만원 이상인 사람을 찾아보면 (화면 11 참조).

인사고과 점수가 60점 이하인 사람을 찾는다 (화면 12 참고).

체크된 사람만을 찾는다.(화면13참고)

체크되지 않는 사람만 찾는다. (화면 14 참고).

생년월일이 61년 2월 1일인 사람을 찾는다 (화면 15 참고).

이름에 '영'자가 포함된 사람을 찾는다 (화면 16 참고).

데이터 찾기

데이터를 찾는 명령으로는 locate를 사용한다.

```
. locate for name='김'
```

이때 김씨로 시작하는 이름을 가진 첫번째 레코드에 위치하게 된다. (화면 17 참고).

데이터의 수정

다음은 입력한 데이터를 수정해보자.

```
. edit 4
```

edit를 입력하면 데이터를 수정할 수 있으며 데이터를 입력할 때와 같은 화면이 나온다. 뒤에 레코드 번호를 적으면 지정한 레코드 번호를 수정할 수 있으며 지정하지 않으면 현재 위치해 있는 레코드 번호를 수정하게 된다.

```
. edit 7 fields code,name,pay
. change 6
```

(edit와 change는 같은 명령이다)

전화면 모드에서 수정을 하지 않고 데이터를 지울려면 replace 명령을 사용한다 (화면 18 참고).

〔화면 4〕

```
.list
Record#  CODE   NAME          PAY  POINT BIRTH    CHECK
      1  000001 윤 택 중   810000  89.00 37.10.10 .T.
      2  000002 이 홍 롱   400000  65.20 61.07.08 .F.
      3  000003 박 준 영   340000  45.00 63.08.06 .T.
      4  000004 채 수 태   600000  87.34 58.12.27 .T.
      5  000005 김 규 봉   390000  77.00 62.12.18 .F.
      6  000006 최 영 숙   396000  76.02 66.05.13 .F.
      7  000007 김 현 진   594000  66.34 65.04.19 .T.
      8  000008 송 진 호   346500  66.60 60.07.31 .T.
      9  000009 최 현 호   356800  56.89 61.02.01 .F.
     10  000010 김 상 녕   787000  66.89 63.02.23 .T.
     11  000011 김 용 석  1200000  84.96 58.09.25 .F.
     12  000012 장 직 순   480000  99.90 56.06.24 .F.
     13  000013 이 정 민   650000 100.00 55.03.21 .F.
     14  000014 전 경 호   570500  33.78 56.11.20 .T.
     15  000015 김 동 군   472500  63.55 57.10.04 .T.
```

〔화면 5〕

```
. goto 4
. display
Record#  CODE   NAME          PAY  POINT BIRTH    CHECK
      4  000004 채 수 태   600000  87.34 58.12.27 .T.
```

〔화면 6〕

```
. go 6
. display
Record#  CODE   NAME          PAY  POINT BIRTH    CHECK
      6  000006 최 영 숙   396000  76.02 66.05.13 .F.
```

〔화면 7〕

```
. 2
. display
Record#  CODE   NAME          PAY  POINT BIRTH    CHECK
      2  000002 이 홍 롱   400000  65.20 61.07.08 .F.
```

〔화면 8〕

```
. list for code > "000012"
Record#  CODE   NAME          PAY  POINT BIRTH    CHECK
     13  000013 이 정 민   650000 100.00 55.03.21 .F.
     14  000014 전 경 호   570500  33.78 56.11.20 .T.
     15  000015 김 동 군   472500  63.55 57.10.04 .T.
```

〔화면 9〕

```
. list for name = "최"
Record#  CODE   NAME            PAY   POINT BIRTH    CHECK
      6  000006 최 영 숙       396000  76.02 66.05.13 .F.
      9  000009 최 현 호       356800  56.89 61.02.01 .F.
     14  000014 전 경 호       570500  33.78 56.11.20 .T.
```

〔화면 10〕

```
. list for name >="전"
Record#  CODE   NAME            PAY   POINT BIRTH    CHECK
      4  000004 채 수 태       600000  87.34 58.12.27 .T.
      6  000006 최 영 숙       396000  76.02 66.05.13 .F.
      9  000009 최 현 호       356800  56.89 61.02.01 .F.
     14  000014 전 경 호       570500  33.78 56.11.20 .T.
```

〔화면 11〕

```
. list for pay>=500000
Record#  CODE   NAME            PAY   POINT BIRTH    CHECK
      1  000001 윤 택 중       810000  89.00 37.10.10 .T.
      4  000004 채 수 태       600000  87.34 58.12.27 .T.
      7  000007 김 현 진       594000  66.34 65.04.19 .T.
     10  000010 김 상 녕       787000  66.89 63.02.23 .T.
     11  000011 김 용 석      1200000  84.96 58.09.25 .F.
     13  000013 이 정 민       650000 100.00 55.03.21 .F.
     14  000014 전 경 호       570500  33.78 56.11.20 .T.
```

〔화면 12〕

```
. list for  point<=60
Record#  CODE   NAME            PAY   POINT BIRTH    CHECK
      3  000003 박 준 영       340000  45.00 63.08.06 .T.
      9  000009 최 현 호       356800  56.89 61.02.01 .F.
     14  000014 전 경 호       570500  33.78 56.11.20 .T.
```

〔화면 13〕

```
. list for  check
Record#  CODE   NAME            PAY   POINT BIRTH    CHECK
      1  000001 윤 택 중       810000  89.00 37.10.10 .T.
      3  000003 박 준 영       340000  45.00 63.08.06 .T.
      4  000004 채 수 태       600000  87.34 58.12.27 .T.
      7  000007 김 현 진       594000  66.34 65.04.19 .T.
      8  000008 송 진 호       346500  66.60 60.07.31 .T.
     10  000010 김 상 녕       787000  66.89 63.02.23 .T.
     14  000014 전 경 호       570500  33.78 56.11.20 .T.
     15  000015 김 동 균       472500  63.55 57.10.04 .T.
```

〔화면 14〕

```
. list for  .not. check
Record#  CODE   NAME            PAY   POINT BIRTH    CHECK
      2  000002 이 흥 룡       400000  65.20 61.07.08 .F.
      5  000005 김 규 봉       390000  77.00 62.12.18 .F.
      6  000006 최 영 숙       396000  76.02 66.05.13 .F.
      9  000009 최 현 호       356800  56.89 61.02.01 .F.
     11  000011 김 용 석      1200000  84.96 58.09.25 .F.
     12  000012 장 직 순       480000  99.90 56.06.24 .F.
     13  000013 이 정 민       650000 100.00 55.03.21 .F.
```

〔화면 15〕

```
. list for  birth=ctod("61.02.01")
Record#  CODE   NAME            PAY   POINT BIRTH    CHECK
      9  000009 최 현 호       356800  56.89 61.02.01 .F.
```

〔화면 16〕

```
. list for  "영"$name
Record#  CODE   NAME            PAY   POINT BIRTH    CHECK
      3  000003 박 준 영       340000  45.00 63.08.06 .T.
      6  000006 최 영 숙       396000  76.02 66.05.13 .F.
```

〔화면 17〕

```
. display
Record#  CODE   NAME            PAY   POINT BIRTH    CHECK
      5  000005 김 규 봉       430000  77.00 62.12.18 .F.
. continue
. display
Record#  CODE   NAME            PAY   POINT BIRTH    CHECK
      7  000007 김 현 진       614000  66.34 65.04.19 .T.
. continue
. display
Record#  CODE   NAME            PAY   POINT BIRTH    CHECK
     10  000010 김 상 녕       807000  66.89 63.02.23 .T.
. continue
. display
Record#  CODE   NAME            PAY   POINT BIRTH    CHECK
     11  000011 김 용 석      1240000  84.96 58.09.25 .F.
. continue
. display
Record#  CODE   NAME            PAY   POINT BIRTH    CHECK
     15  000015 김 동 균       492500  63.55 57.10.04 .T.
. continue

End of LOCATE scope
```

〔화면 18〕
```
. replace pay with 455000
. replace point with point+5 for point >= 70
. replace birth with birth+20
. replace all pay with pay+10000
. bonus=20000
. replace all pay with pay+bonus
```

replace 명령시 조건이나 all을 사용할때 외에는 현재의 레코드를 대치하게 되므로 미리 레코드 번호를 지정해 주어야 한다.

다른 레코드와 비교하면서 데이터를 수정하는 방법으로는 browse가 있다.

```
. browse
. browse fields name,point
```

데이터 지우기

데이터를 입력하면 삭제할 필요가 있다. 데이터를 삭제하는 명령으로는 delete라는 명령을 사용한다.
(화면 19 참고).

위와 같이 데이터를 지운다해도 레코드의 첫번째에 삭제 표시만을 하고 실제로는 지우지 않는다.

삭제 표시한 데이터를 실제 지운 것처럼 작동하게 하기 위해서는

```
. set delete on
```

이라 지정해 놓으면 된다.(화면20참고)

모든 데이터를 지울때는

```
. delete all
```

이라 명령하면 되고, 다른 방법으로

```
. zap
```

이라 입력하여도 모든 데이터를 지울 수 있으나 zap을 사용하면 다시 살릴 수 없게 된다.

데이터 살리기

데이터를 지웠어도 실제로는 지우지 않고 삭제 표시만을 했으므로 다시 살릴 수가 있다. 지웠던 데이터를 다시 살리기 위해서는 recall을 사용한다.
(화면 21 참고).

지웠던 모든 데이터를 다시 살리기 위해서는

```
. recall all
```

이라 입력한다.

데이터의 완전 삭제

데이터를 1개 지울 때마다 빈칸을 다음 레코드에 옮기고 그자리에 다음 레코드를 옮긴다면 데이터 화일을 새로 만드는 작업과 같은 효과가 된다.

어떠한 데이터를 처리하는 프로그램도 데이터를 지움으로써 그 공백은 항상 남게 된다.

디베이스에서는 삭제한 데이터를 공백으로 만들지 않고 삭제 표시만을 하여 나중에 다시 재생할 수 있게 했으며 완전히 제거하는 명령을 따로 만들어 사용하게 되어 있다.

pack이란 명령은 삭제 표시한 레코드를 완전히 제거하는 것으로 실제로는 삭제표시 된 레코드를 제외한 레코드를 동일한 화일명으로 복사하는 작업을 하게 된다.

데이터가 많을수록 시간이 소요되므로 이 명령의 사용은 많은 데이터를 사용하는 프로그램내에서는 가급적 피하는 것이 좋다. pack을 사용하면 지웠던 레코드를 다시 살릴 수가 없다.

데이터의 계산

입력된 데이터베이스 화일을 계산하는 방법을 알아보자.

데이터의 갯수를 알아보는 명령으로는 count를 사용한다.

결과를 변수로 받아들이기 위해서는 to 〈변수명〉을 뒤에 붙이면 새로운 변수가 생긴다.

```
. count
    15 records
. count for name=[박]
    1 record
. count for name=[최]
    2 records
. count for name=[김]
    5 records
. count for name=[김] to kimcnt
    5 records
. ? kimcnt
        5
```

수치 필드의 합계계산을 위해서는 sum을 사용하며 이의 평균 값을 계산하기 위해서는 average를 사용한다.

```
. sum point
    15 records summed
   point
 1078.47
. sum pay
    15 records summed
      pay
 8393300
. sum pay for name=[김]
     5 records summed
      pay
 3443500
. sum pay for name=[김]  to kimsum
     5 records summed
      pay
 3443500
. ? kimsum
 3443500

. average pay
    15 records averaged
   pay
559553
. average point
    15 records averaged
point
71.90
```

```
. average pay for name=[김]  to kimavr
     5 records averaged
   pay
688700
 ? kimavr
   688700
```

변수의 조작

지금까지 데이터베이스 화일을 작성하고 데이터의 입력, 수정, 삭제등을 알아보았다. 지금까지 작성되었던 변수를 알아보자.

현재의 기억 변수가 있는가를 알아보는 명령은

```
.list memory
```

이다. 또한 이 변수를 지우고자 할때는

```
. release <변수명>
```

을 사용한다.

또 kim으로 시작하는 변수만을 지우고자 할 때는

```
. release all like kim*
```

kim으로 시작하는 변수만을 남기고 다른 변수를 지우고자 할 때는

```
. release all except kim*
```

모든 변수를 지우고자 할 때는

```
. release all
```

또는

```
. clear memory
```

라고 명령하면 된다.

또 변수의 내용을 화일로 저장했다

〔화면 19〕

```
. delete for recno()=5
     1 record deleted
. delete for point<=70
     8 record deleted
. list
Record#  CODE   NAME          PAY  POINT BIRTH    CHECK
      1  000001 윤 택 중    810000  89.00 37.10.10 .T.
      2 *000002 이 홍 룡    400000  65.20 61.07.08 .F.
      3 *000003 박 준 영    340000  45.00 63.08.06 .T.
      4  000004 채 수 태    600000  87.34 58.12.27 .T.
      5 *000005 김 규 봉    390000  77.00 62.12.18 .F.
      6  000006 최 영 숙    396000  76.02 66.05.13 .F.
      7 *000007 김 현 진    594000  66.34 65.04.19 .T.
      8 *000008 송 진 호    346500  66.60 60.07.31 .T.
      9 *000009 최 현 호    356800  56.89 61.02.01 .F.
     10 *000010 김 상 녕    787000  66.89 63.02.23 .T.
     11  000011 김 용 석   1200000  84.96 58.09.25 .F.
     12  000012 장 직 순    480000  99.90 56.06.24 .F.
     13  000013 이 정 민    650000 100.00 55.03.21 .F.
     14 *000014 전 경 호    570500  33.78 56.11.20 .T.
     15 *000015 김 동 균    472500  63.55 57.10.04 .T.
```

〔화면 20〕

```
. list
Record#  CODE   NAME          PAY  POINT BIRTH    CHECK
      1  000001 윤 택 중    810000  89.00 37.10.10 .T.
      4  000004 채 수 태    600000  87.34 58.12.27 .T.
      6  000006 최 영 숙    396000  76.02 66.05.13 .F.
     11  000011 김 용 석   1200000  84.96 58.09.25 .F.
     12  000012 장 직 순    480000  99.90 56.06.24 .F.
     13  000013 이 정 민    650000 100.00 55.03.21 .F.
```

〔화면 21〕

```
. recall for recno() < 5
. recall for recno() = 15
. list
Record#  CODE   NAME          PAY  POINT BIRTH    CHECK
      1  000001 윤 택 중    810000  89.00 37.10.10 .T.
      2  000002 이 홍 룡    400000  65.20 61.07.08 .F.
      3  000003 박 준 영    340000  45.00 63.08.06 .T.
      4  000004 채 수 태    600000  87.34 58.12.27 .T.
      6  000006 최 영 숙    396000  76.02 66.05.13 .F.
     11  000011 김 용 석   1200000  84.96 58.09.25 .F.
     12  000012 장 직 순    480000  99.90 56.06.24 .F.
     13  000013 이 정 민    650000 100.00 55.03.21 .F.
     15  000015 김 동 균    472500  63.55 57.10.04 .T.
```

〔화면 22〕

```
. set alternate to history
. use good
. list
. set alternate on
. list for recno() > 10
. list code=[000001]
. set alternate off
. list for pay > 500000
. list memory
. set alternate on
. go 10
. display
. 12
. display
. display memory
. list fields name,"-----",point,'-----',birth,[-----],code
. close alternate
. type history.txt
```

가 언제라도 다시 사용할 수 있는 방법이 있다.

변수를 화일로 저장하는 명령은 다음과 같다.

.save to 〈화일이름〉

변수를 화일로 저장할때 화일이름의 확장자는 확장자 이름을 주지않으면 자동으로 .mem이라는 확장자를 붙여준다.

화일로 저장할 때에도 변수를 지울때처럼 like와 except를 사용하여 변수를 저장할 수 있다

. save to 〈화일이름〉 all like kim*
. save to 〈화일이름〉 all except kim*

변수를 화일에서 다시 불러내는 명령은 .restore from〈화일이름〉이다. 변수를 화일에서 불러내면 기존에 있던 변수는 모두 지워지고 화일에서 불러낸 변수만 남게 된다.

기존에 있던 변수와 새로운 화일에서 불러낸 변수가 공존하게 하기 위해서는 다음과 같이 명령해야 한다.

. restore from 〈화일이름〉 additive

작업의 기록

다음은 지금까지 작업을 복습하는 의미로 아래의 순서대로 차례로 입력해 보길 바란다. (화면 22 참고).

여기까지 차례로 입력해 보면 set alternate 라는 명령이 무엇이라는 것을 저절로 알게 될 것이다.

set alternate to 〈화일이름〉으로 지정한 화일에 set alternate on 상태인 경우만 화일안으로 기록되고, set alternate off인 경우는 기록되지 않는다. 그리고 마지막에 반드시 close alternate 를 쳐줌으로 화일에 저장을 끝낼 수가 있다.

이 화일은 텍스트 화일이므로 독자가 사용할 수 있는 워드프로세서로 편집을 할 수 있다.

set alternate to 〈화일이름〉에서 화일이름에 확장자를 붙이지 않으면 디베이스는 자동으로 .txt란 확장자를 붙여준다.

이번호에서는 데이터베이스 화일의 작성과 간단한 이용을 해보았다. 다음호에서는 데이터베이스 화일의 정렬과 복수개의 데이터베이스 화일의 사용을 알아보기로 하자.

본지에서는 꼭 알아야 할 사항만을 소개하고 초보자의 길잡이의 역할을 하는데 의의가 있으므로 한가지 명령의 세세한 설명은 가급적 피했다. 좀더 자세히 알고자 하는 독자는 아래 제시한 명령어를 복습을 겸해 매뉴얼을 구입해 익혀두면 훨씬 깊은 이해를 할 수 있을 것이다. PC

〈이번호에서 나왔던 명령어와 함수〉

create	set carry on/OFF	sum
modify structure	browse	average
list structure	replace with	release
use	locate	save to
list	continue	restore from
display	delete	set alternate to
[goto] 레코드번호	zap	set alternate on/OFF
append	recall	recno()
edit / change	pack	ctod(c)
set delete on/OFF	count	

에러처리루틴의 구성

대부분의 양호한 베이직 프로그램들은 에러가 발생할 때마다 베이직으로 돌아가거나 파괴되는 것으로 부터 그들을 보호하는 에러 트래핑(Error Trapping)을 가지고 있다. 프로그램내의 ON ERROR GOTO문은 에러를 인식하고 적당한 행동을 취하는 그러한 역할을 한다.

여기에 소개하는 에러처리 루틴은 완벽한 것은 되지 못하지만 어떻게 에러가 발생하며 어떠한 과정을 거쳐서 이러한 에러를 대처하는지를 잘 말해주고 있다. 따라서 이 프로그램을 잘 이용하면 간단히 각자의 프로그램에 에러처리 루틴을 설치할 수 있을 것이다.

에러의 원인은 사람이다.

간단한 에러라 할지라도 프로그램을 파괴시키거나 심각한 문제를 일으킬 수 있다. 잘못된 FOR NEXT문, 신텍스 에러, 그리고 존재하지 않은 라인에 점프하는 GOTO와 같은 프로그래밍 에러는 방지하기가 대단히 어렵다. 이것은 보통 사용자의 잘못된 입력에 의해 발생되는 것이다. 따라서 그와 같은 에러들은 허용되는 입력의 범위를 정확하게 정의하는 프로그램 문으로 방지할 수 있다. 이러한 형태의 에러들에 대한 에러 트랩은 당연히 필요하지 않다.

그러나 외부적인 런 타임 에러들은 데이터를 입력하거나 프로그램을 수행하는 시간을 소비한 후에 발생하기 때문에 특히 귀찮으며 프로그램에 미치는 영향이 매우 심각하다고 할 수 있다. 이런 에러로 프로그램이 파괴되었을 때는 모든 노력이 헛수고가 된다. 어떻게 하여 프로그램을 헛되게 하는지를 생각해 보자.

예를 들어, 재미있는 게임을 하면서 시간을 소비한다든가, SAVE 명령을 시도할 때 디스크가 완전히 차있기 때문에 프로그램을 파괴시키는 경우 등등.

리스트 1의 에러 트랩을 사용하기 위해서는 프로그램을 입력한 후 그것을 다음과 같은 명령을 삽입하여 ASCII 화일로서 세이브한다.

```
SAVE"ERRTRAP",A
```

프로그램의 세이브가 끝난 후에는 머지(Merge)하고 싶은 프로그램을 로드한 후

```
MERGE"ERRTRAP"
```

을 타이핑하고 리턴키를 누름으로써 그 프로그램에 에러 트랩을 머지시킬 수 있다.

〔리스트 1〕 ERRTRAP.BAS 프로그램

```
1 ON ERROR GOTO 65000
65000 IF ERR=24 OR ERR=25 OR ERR=27 THEN INPUT "Device
      not ready. Fix and press ENTER.",DUMMY$:RESUME
65002 IF ERR=53 THEN INPUT "File not found. New file
      name";F$:RESUME
65004 IF ERR=57 OR ERR=72 THEN INPUT "Bad disk. Repl
      ace and press ENTER.",DUMMY$:RESUME
65006 IF ERR=61 OR ERR=67 THEN INPUT "Disk full. Rep
      lace and press ENTER.",DUMMY$:RESUME
65008 IF ERR=64 THEN INPUT "Bad file name. New file
      name";F$:RESUME
65010 IF ERR=70 THEN INPUT "Disk write protected. Ch
      ange and press ENTER.",DUMMY$:RESUME
65012 IF ERR=71 THEN INPUT "Disk not ready. Fix and
      press ENTER.",DUMMY$:RESUME
65014 ON ERROR GOTO 0
```

트랩작업은 어떻게 이루어지나.

여러가지 외부의 런 타임 에러들에 익숙해지기 위해서, 그리고 프로그램이 그 에러들 중의 하나를 만날 가능성을 알기 위해서는 리스트내의 라인들에 관해 어느 정도 알아야 한다. 각 라인들이 어떻게 작업하는지를 간단히 알아보자.

라인 1은 라인 65000에서 시작하는 에러 트래핑 코드를 컴퓨터에 보내 트랩을 실행시킨다. 대부분의 프로그램이 라인 1이나 65000 이상의 라인을 갖고 있지 않기 때문에 그 라인들을 리넘버링(Renumbering) 없이 머지(Merge)시킬 수 있다. 그러나, 만일 프로그램이 CLEAR문을 포함하고 있다면, ON ERROR GOTO라인을 CLEAR문 다음에 오도록 하기 위하여 그 라인에 리넘버를 해야 한다. CLEAR문은 에러 트래핑을 해제시키는 역할을 한다.

라인 65000은 컴퓨터가 준비되지 않은 주변장치를 사용할려고 시도할 때 발생하는 에러를 포착한다. 프린터가 켜지지 않았다든지, 또는 모뎀으로 통신하는 동안 폰 접속이 유실되었다든지, 또는 상대편에서 컴퓨터가 응답하지 않았다든지 등의 에러를 찾아낸다.

라인 65000은 "Device not ready"와 같은 메세지를 프린트하며, 유저로 하여금 그 디바이스를 켜고 프로그램을 떠난 곳으로 되돌아 가도록 프롬프트한다. Error 24(Device Timeout)는 베이직이 이미 지정된 시간내에 I/O 디바이스로부터 정보를 받아들이지 않았다는 것을 뜻한다. Error 25(Device Fault)는 프린터 또는 인터페이스카드에 뭔가 잘못이 있다는 것을 의미한다. Error 27(Out of Paper)은 프린터에 용지가 없거나 프린터에 또 다른 문제가 있는 것을 나타낸다.

라인 65002는 컴퓨터가 존재하지도 않는 화일을 로드할려고 시도할 때 발생하는 에러를 포착한다. 그것은 "File not found. New file name?"과 같은 메시지를 프린트한다. 이때 새로운 화일명을 입력하면 프로그램은 그 화일을 로드할려고 시도할 것이다.

라인 65002는 원래의 프로그램이 프롬프트된 화일명을 변수 FS에 저장한다는 가정을 한다. 만일 프로그램내에 다른 변수를 사용한다면, 그 변수를 에러 트랩의 라인 65002에 FS 대신에 삽입해야 한다. 프로그램을 통하여 화일명에 대한 똑같은 변수를 사용해야 한다.

새로운 화일명을 찾기전에 유저로 하여금 디스크상의 화일 리스트를 보도록 할 수 있다. INPUT문 대신 다음과 같이 라인 65002를 바꾼다.

```
PRINT "File not found. Files on disk are:":
FILES:INPUT"New file name";F$
```

에러 트랩에 이러한 라인을 추가할 때는 약간의 문제점이 발생할 수도 있다. 즉, 디스크상에 화일이 존재하지 않을 경우 프로그램은 에러 트랩내에서 File Not Found 에러를 만나게 될 것이며 파괴된다든지 또는 FILE 명령은 디스크상에서 데이터화일이 아니라 모든 화일을 보여준다든지 하는 점이다.

라인 65004는 Bad Disk 에러를 포착하는데, 이것은 포맷되지 않거나 배드 디렉토리(Bad Directory)를 갖는 화일상에서 세이브나 로드를 하려고 컴퓨터가 시행할 때 일반적으로 발생하는 에러이다. 유저는 그 디스크를 교체하고 운영으로 다시 되돌리도록 리턴키를 누를 수 있다.

라인 65006은 유저가 풀(Full)상태에 있는 디스크상에 데이터를 세이브하려고 시도할 때 "Disk full"이라는 메세지를 프린트한다. 그 라인은 디스크를 교체하고 운영이 계속되도록 리턴키를 누른다. 보통 오리지널 프로그램은 SAVE 명령을 반복하지 않을 것이며, 그래서 유저는 다시 그 프로그램에 들어가야 한다(어떤 경우, 베이직은 이러한 에러로부터 복원되지 못하고 프로그램은 파괴된다. 그러나 부분적인 보호로서 라인 65006을 유지하는 것도 좋다.).

라인 65008은 컴퓨터가 화일을 비적절한 화일명으로 오픈하려고 시도할 때 발생하는 에러를 포착한다. 보통 프로그램은 유저에게 데이터를 세이브하려는 상황에서 화일명을 입력하라고 물어오며 유저는 아스테리스크(*)와 같은 비적절한 캐릭터를 포함 시킬 수도 있다. 이러한 라인은 유저에게 새로운 화일명을 입력하라고 물어오고 그래서 화일을 오픈하는 것을 다시 시도한다.

라인 65002에서 처럼 라인 65008은 화일명이 FS에 저장된다고 가정한다. 만일 프로그램이 다르다면, 이 라인에 FS를 새로운 변수로 바꿔야 한다. 에러 64는 보통 새로운 화일을 세이브하려고 시도할 때—기존 화일을 로드할 때가 아니라—발생하기 때문에 라인 65008에 FILES 명령을 포함시키는 데는 이유가 거의 없다.

라인 65010은 라이트-프로텍트된 디스크상에 데이터를 세이브하려고 시도할 때 발생하는 에러를 포착한다. 이 라인은 유저에게 디스크를 교체하여 원래의 운영으로 되돌아 가도록 리턴키를 누를 기회를 준다.

라인 65012는 유저가 입력 또는 출력을 위해 화일을 오픈할 때 발생하는 Disk Not Ready 에러를 포착한다. 드라이브 도어가 열려 있다든지, 드라이브상에 디스크가 없을 때이다. 다른 라인들과 마찬가지로 라인 65012는 유저로 하여금 그 상태를 수정하여 원래의 운영으로 다시 시작하도록 리턴키를 누르게끔 한다.

ERRTRAP.BAS는 프로그래밍이나 런 타임 에러를 위해 이미 디버그된 프로그램내에서 발생하기 쉬운 모든 에러들을 처리한다. 만일 에러가 런 타임 형태라면 라인 65014는 트래핑을 지우고 프로그램을 에러 메세지로 끝낸다.

결론

에러 메세지를 화면상의 원하는 위치에 표시할 수 있다면 디버그시 매우 편리할 것이다. 그렇게 하기 위해서는 에러 메세지를 디스플레이하는 각각의 INPUT문 앞에 LOCATE 명령을 삽입하면 된다.

또한, 에러 메세지 시작에 커서를 놓을 수도 있고, 유저가 리턴키를 누른 후 그 메세지를 지우기 위한 공간의 스트링을 프린트 한다. 이렇게 하기 위해서는 에러 트랩 각 라인의 RESUME문 전에 LOCATE문을 놓는다.

예를 들어 위에서 두가지의 변형을 하기 위해서는 라인 65012를 다음과 같은 것으로 바꿔야 한다.

```
65012 IF ERR = 71 THEN LOCATE
25,1:INPUT "Disk not ready. Fix and press
```

```
ENTER.",DUMMY$:LOCATE 25,1:PRINT
TAB(39):RESUME
```

RESUME문을 변형하여 CLOSE문을 추가함으로써 Disk Full 트랩을 좀더 신뢰성이 있도록 기회를 만들 수 있다. 예를 들어, 라인 65006을 다음과 같은 것으로 바꿔서

```
65006 IF ERR = 61 OR ERR = 67 THEN INPUT
"Disk full. Replace and press ENTER.",
DUMMY$:CLOSE:RESUME 150
```

와 같이 하면 유저로 하여금 오리지날 프로그램의 라인 150으로 되돌아 가게 하는데, 여기서 라인 150은 화일을 오픈하려고 원래 시도되었던 OPEN문을 포함한다. 그와 같은 변경은 ERRTRAP.BAS를 전송면에서 좀더 뒤떨어지게 한다는데 유의하라.

결론적으로 어떤 프로그램은 부가적인 에러 트래핑을 필요로 한다. 예를 들어 모뎀을 통해 통신하는 프로그램들은 패리티 에러와 전기통신에서 유일한 다른 문제들을 체크해야 한다. 그러한 점들은 일반화된 에러 트랩에 대해서는 너무 복잡하다. 이러한 제한을 갖고 있지만, ERRTRAP.BAS는 최소의 작업으로 프로그램을 안전하게 보호해 줄 것이다. PC

디지탈 필터의 구현

최승욱/태성전산

지난호에서는 아날로그 디지탈 변환기의 제작에 관하여 알아보았다. 이번 호에서는 지난호에 이어 제작된 ADC(아날로그 디지탈 변환기)를 기본으로 하여 데이터를 읽는 방법에 대하여 설명하고자 한다.

이미 설명한 것과 마찬가지로 아날로그디지탈 변환기는 주로 외부에서 발생한 연속적인 물리량을 이산적인 디지탈 신호로 변환(이를 양자화라고 한다)하여 처리할 수 있도록 해주는 장치이다.

그런데 이와같이 읽혀진 데이터에는 우리에게 필요한 것이 있고 또한 읽혀서는 안됨에도 불구하고 읽혀지는 데이터도 있다. 그래서 읽지 말아야할 데이터를 제거해야 하는 문제가 발생하는데 이때는 보통 필터를 사용하여 해결한다.

MS-DOS나 데이터베이스를 사용하다 보면 필터란 말을 자주 듣게 되는데 ADC를 사용하는 경우에도 역시 필터(filter)가 필요하다. 필터는 필요한 것과 필요하지 않은 것을 어떤 형식으로 지정하여 원하는 것만을 골라주는 것이다.

그러면 이렇게 읽혀진 데이터를 어떻게 처리할 것인가?

이미 만들어진 ADC의 입력에 일정한 전압을 걸고 계속해서 읽어보면 비교적 정확히 데이터를 읽지만 가끔 엉뚱한 코드가 나타나는 현상을 발견하게 될 것이다. 단순히 일정한 순서에 의해 데이터를 읽을 경우 이런 현상은 문제를 일으키는 원인이 되는데 결국 이러한 문제는 소프트웨어나 하드웨어로써 해결할 수 있다.

앞에서 이미 설명한 대로 읽혀져야 할 데이터와 그렇지 않아야 할 데이터를 구별하는 것이 가장 중요한 일이라 할 수 있는데 이러한 것의 완벽한 구별은 거의 구현하기 어렵기 때문에 실제로는 읽혀질 데이터의 형태를 대강 예상하여 그것을 바탕으로 데이터를 읽어야 한다. 물론 여기서 말하는 읽혀질 데이터란 측정할 참데이터이고 읽혀지지 말아야 하는 것은 측정환경과 측정오차에 의해 발생하는 임펄스나 전역에 걸쳐서 존재하는 백색 잡음과 같은 노이즈(noise)성분을 의미한다.

결국 여기서 설명하는 데이터 분석이란 이와같이 노이즈성분 제거라는 의미에서 설명될 수 있다고 하겠다.

노이즈의 제거는 ADC의 입력단에 있는 전치증폭기에 의해서 크게 개선될 수 있다. 즉 입력단에 능동필터를 설치하여 입력부에서부터 신호에 혼합된 노이즈성분을 제거하는 것이다. 또한 이와같은 전자회로로 구성된 필터는 소프트웨어적으로도 구현할 수 있으며 이를 디지탈필터라고 한다.

한편 읽혀진 데이터에 주파수 성분이 개입되었을 경우 다른 분석법에 의하여 분석될 수 있는데 푸리에(Fourier) 변환이나 왈쉬(Walsh) 변환같은 방법에 의하여 데이터를 분석하는 경우도 있고 필요에 의해서는 읽혀진 이산적인 데이터를 예상되는 형태의 수식으로 변환하는 커브피팅(Curve Fitting)이란 방법도 있다.

앞으로 설명한 부분도 노이즈의 제거방법과 리니어 커브피팅 및 그밖의 디지탈 필터의 구현방법에 관한 것들인데 이러한 것들은 실제에 있어서 다분히 이론적일 수 있으나 여기서는 되도록이면 수식을 피하고 실제로 컴퓨터를 사용하여 구현하고 그 실용적인 재미를 제공하기 위하여 실현의 예와 함께 설명하겠다.

(1) 노이즈의 제거방법

• 능동필터 구성에 의한 방법

병법에 적을 알고 나를 알면 백번싸워서 백번 다 이긴다는 말이 있다. 갑자기 왜 이런 말을 하는가 하면 실제로 노이즈의 혼입은 전자회로의 설계시 진정한 적이라 할 수 있기 때문이다. 어떤 스펙에 의해 회로를 설계하는 경우 모든 조건을 만족하는 노이즈제거기를 만드는 것은 거의 실현하기 어렵다.

실제로 노이즈는 전대역에 걸쳐서 존재하며 이를 모두 제거하려다가는 신호마저 제거하게 되어 소용이 없게 된다. 따라서 측정하려는 신호의 성질을 알아야 하며 이를 바탕으로 하여 필터를 설계한다.

우리 주위에서도 비교적 성능이 우수한 필터를 볼 수 있는데 고급 오디오 시스템의 그래픽 이퀄라이저도 사실상 대역필터이며 실제로 측정신호가 가청주파수내의 신호성분이면 이를 사용하여 데이터를 읽을 수도 있다.

설명이 약간 빗나간 것 같은데 다시 필터설계에 돌아와서 보면 이와같은

필터에는 몇가지 종류가 있으며 그것은 다음과 같다.

① 대역필터(Band Pass Filter)
② 대역소거필터(Band Eliminate Filter)
③ 저역필터(Low Pass Filter)
④ 고역필터(High Pass Filter)

그림 1에 각 필터의 특성을 보였다. 그림 1에서 알 수 있듯이 대역필터는 말그대로 특정주파수 구역만 통과시키는 필터이다. 반대로 대역소거필터는 특정주파수 구역에서만 신호를 감쇄시키는 필터이고 저역필터와 고역필터는 낮은 주파수와 높은 주파수만을 각각 통과시키는 특성을 갖는 필터이다.

그러면 필터의 설계란 무엇인가? 그림 1과 같이 필터의 설계는 높은 주파수, 낮은 주파수를 전자회로의 시정수로써 구현하고 통과시킬 주파수와 통과시키지 말아야 할 주파수를 정하고 이를 역시 시정수로써 구성하는 것이다. 이것이 결국 필터설계의 요체가 되고 또한 디지탈필터의 구성에 있어서도 마찬가지이다. 물론 여기서는 필터의 골격설계는 이미 지원된 것을 전제로 한다.

실제 설계의 경우 이런 식으로 진행하는 것이 비교적 빠르고 안전한 설계 지침이라 할 수 있다.

여기서는 몇가지의 필터의 기본골격을 제시하고 이것의 시정수를 계산하는 법과 실제적 능동필터의 구현을 보인다. **그림 2**에 그러한 구현을 나타내었다.

• 디지탈필터의 구성법

앞에서 전자회로의 구성에 의하여 필터를 설계한 것과 마찬가지로 아날로그적인 동작을 컴퓨터에게 모의적으로 동작시키게 되면 즉, 시뮬레이션하면 그것이 곧 디지탈필터가 된다. 일반적인 디지탈필터는 입력되는 신호가 이산적인 데이터의 열이고 따라서 출력 역시 이산적인 데이터의 열로써 나타난다. 이의 모델링은 **그림 3**과 같다.

이때 X(z)는 ADC로부터의 입력신호의 열이고 H(z)는 프로그램으로써 구성되어져야 할 부분이다. 물론 입력이 이산적이므로 출력 역시 이산적인 데이터의 열이 된다.

이를 수식으로 표현하면 다음과 같이 입력은

$$\mathbf{x(n)=\sum_{n=-\infty}^{\infty} x(m)d(n-m)}$$

라고 표시할 수 있고, 출력y(n)은 다음식으로 주어진다.

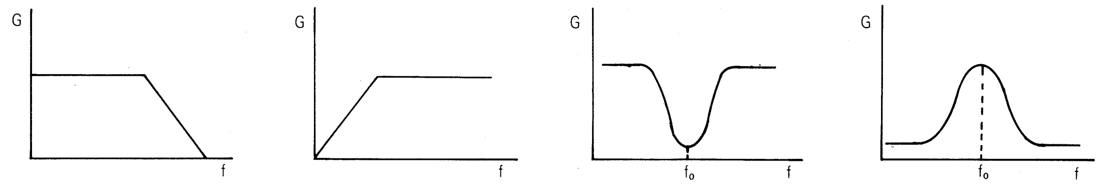
$$\mathbf{y(n)=\sum_{n=-\infty}^{\infty} x(m)h(n-m)}$$

여기서 h(n-m)은 임펄스응답이다. 이때 이 임펄스응답이 시간에 대하여 불변이라고 하면 이를 간단히 다음과 같이 쓸 수 있다.

$$\mathbf{y(n)=x(n)*h(n)}$$

라고 쓰는데 이때 ∗는 곱하기 표시가 아니고 합성 즉 콘볼루션이라는 새로운 연산자로써 표시할 수 있다. 결국 이를 실제로 구현하기 위해서는 위의 임펄스응답을 어떠한 형식으로 만들어

〔그림 1〕 각종 필터의 특성

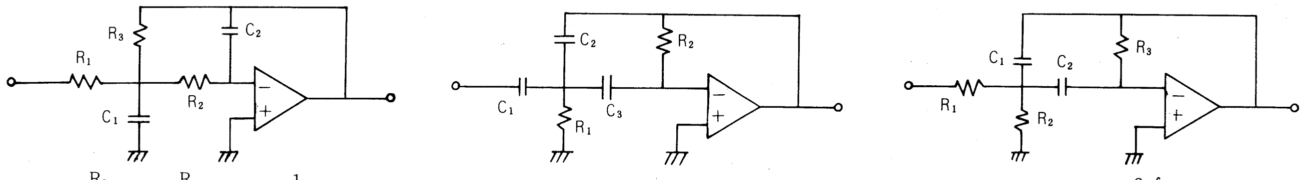


(a) 저역필터　(b) 고역필터　(c) 대역소거필터　(d) 대역필터

여기서 G : 이득(Gain)
f : 진동 주파수
f_0 : 공진 주파수

〔그림 2〕 각종 필터의 모습과 계산법

(1) 저역필터　(2) 고역필터　(3) 대역필터



(1) 저역필터

$R_1=\frac{R_3}{G}$　$R_2=\frac{R_3}{G+1}$　$R_3=\frac{1}{\sqrt{2}K}$

여기서 $K=2\pi f_c C_2$
$C_1=2(G+1)C_2$ 이고
f_c : 차단주파수
데이터는 G : 이득
C_2의 용량이 주어진다.

(2) 고역필터

$C_1=GC_2$　$C_3=C_1$　$K=2\pi fC_1$

$R_1=\frac{\sqrt{2}G}{K(2G+1)}$　$R_2=\frac{2G+1}{\sqrt{2}K}$

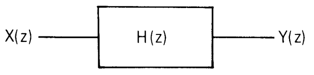
(3) 대역필터

$C_1=C_2=C$　$a=\frac{2\pi f}{Q}$　$b=(2\pi f)^2$

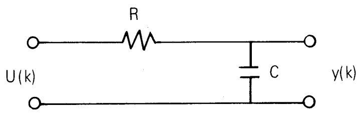
$R_1=\frac{1}{aC|G|}$　$R_3=\frac{2}{aC}$　$R_2=\frac{1}{C\left[\frac{2b}{a}-a|G|\right]}$

단, $Q\leqslant 10$
여기서 Q는 Quality factor의 약어

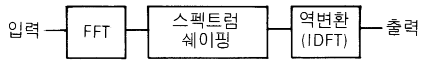
〔그림 3〕 디지탈 필터의 모델링



〔그림 4〕 저역 필터



〔그림 5〕 FFT를 통한 필터링의 블록 다이어그램



(*IDFT=Inverse Discreate Fourier Transform
*FFT=Fast Fourier Transform

주느냐에 따라서 여러가지 특성을 갖는 필터를 구성할 수 있는 것이다.

위의 경우h(n)의 길이가 유한한가 무한한가에 의해서 FIR(Finite Impulse Response)과 IIR(Infinite Impulse Response) 필터로 구별된다.

여기서는 유한한 경우인 FIR필터를 구현한다.

FIR필터는 입력의 신호열을 x(n), 출력열을 y(n)이라 하면 출력식은 다음과 같이 표현된다.

$$\begin{aligned} y(n) &= h(0)x(n)+h(1)x(n-1) \\ &\quad +h(2)x(n-2)+\cdots+h(n-1) \\ &\quad x(1)+h(n)x(0) \end{aligned}$$

위의 식을 보면 어떤 식으로 프로그래밍을 해야할 것인가를 떠올릴 수 있을 것이다. 물론 임펄스응답식을 구했을 경우를 전제한 것이다. 위의 식은 앞에 설명한 합성에 관한 정의가 될 수도 있다.

여기서 위의 설명 내용을 실제로 컴퓨터상에서 구현하려면 먼저 필터의 임펄스응답이 필요하다. 이러한 응답을 구하기 위해서는 필요한 필터의 기본골격을 구성하고 그것에 대한 미분방정식의 해를 구하여 이를 메모리에 저장하여 위의 식에 대입하여 실현할 수 있다.

여기서는 비교적 일반적이고 주파수가 낮은 분야에 사용될 수 있는 저역필터의 경우를 구현하고 이의 임펄스응답은 푸는 과정을 생략하고 결과만의 사용에 대해서 설명한다. 여러분은 지난호에서 비교적 잡음에 강하고 일반적 용도에 맞는다는 이중적분형 ADC를 기억할 것이다. 이러한 이중적분형 ADC도 사실상 입력의 적분기에서 저역필터의 작용을 함으로써 이러한 필터의 작용을 함께하게 되는 것이다. 즉 적분기는 저역필터와 똑같은 동작을 하기 때문에 저주파의 노이즈를 제거하는 필터효과를 낼 수 있는 것이다.

• **간이 형식에 의한 디지탈필터의 구현**

그러면 앞의 설명에 따라 필터를 설계하고 이를 프로그램으로 작성해 보자. 우선 저역필터의 일반적 형태를 그림 4에 나타내었다. 그림 4를 보면 저항에 캐퍼시터가 병렬로 연결된 형태의 회로로 구성된다. 여기서 시정수는 저항과 캐퍼시터의 곱으로 정해지고 이것은 필터의 커트오프 주파수를 결정하게 된다.

그리고 이러한 물리적 회로를 컴퓨터를 사용하여 이를 대신 처리하게 하려면 임펄스응답을 구해야 한다. 만일 임펄스응답이 지나치게 복잡하게 나온다면 실제 프로그램으로 구현이 되어도 계산에 많은 시간이 소비되기 때문에 실시간으로 처리하는데 문제가 발생하므로 가능한 간단한 구조의 회로가 바람직하다. 그림 4의 임펄스 응답은 다음과 같다.

$$H(s)=\frac{1}{RCs+1}$$

〔여기서 시정수 T=RC h(t)=exp(−t/RC)〕

이를 풀면 디지탈필터의 출력은

$$y(k)= a*u(k)+\exp(-a)*y(k-1)$$

〔여기서 $a=\frac{T}{RC}$이고, T는 샘플링 주기 k=0, 1, 2, ……, n〕

이고 이때의 커트오프 주파수는 다음과 같다.

$$f_c=\frac{1}{2\pi\sqrt{RC}}$$

이제 모든 준비는 다 됐다. 만일 R=10K 이고 C=10mF이면 커트오프 주파수는 약 0.5Hz이고 따라서 거의 직류적 특성을 갖게 되어 0.5Hz 이하만 통과시킨다. 여기서 임펄스 응답h(t)는 배열변수로 미리 갖게 하여 입력된 신호와 합성을 취하면 그 결과가 곧 필터링된 출력이 된다. 리스트 1은 이의 예제 프로그램이다.

• **푸리에 변환 해석법에 의한 경우**

앞에서 설명한 필터의 경우는 그 입력신호가 맥동이나 전 신호값과 현재 신호값의 차이가 설정치 이상인 경우의 신호를 생각하여 설계한 경우이고 지금부터 다루어지는 내용은 주파수 성분을 함유한 신호에 관한 경우가 된다.

푸리에변환은 신호처리분야나 전기공학등에서는 대단히 중요한 부분으로 되어 있고 실제로 이것을 응용한 분야도 매우 넓다. 예를 들면 팝송을 들을 때 들리는 전자악기의 신호합성 즉 여러가지 소리를 내는 것도 푸리에이론의 응용부분이다. 또한 문자의 인식이나 화상, 음성인식 등에도 이것의 응용이 핵심이 된다.

이와같이 푸리에변환은 여러분야에 걸쳐서 사용되는 이론인데 여기서는 필터링의 응용으로써 소개하겠다.

푸리에 변환은 간단히 설명하면 어떤 주기적인 신호를 사인파만을 이용하여 표시할 수 있다는 이론이다. 즉 어떤 주기신호도 사인항의 급수로 전개할 수 있다는 것이다.

그림 5는 고속 푸리에변환 알고리즘(FFT)을 통한 필터링의 블록다이어그램이다.

그림 5를 간단히 설명하면 푸리에변

〔리스트 1〕 FIR 필터

```
PROGRAM FIR_FILTER_REALIZATION;

USES CRT;
VAR  U,Y,PY,T:REAL;

BEGIN
T:=1;   {Time constant}
PY:=0;  {Initial value}

REPEAT
   U:=5+RANDOM(3);     {Generate pseudo random input signal}
   Y:=T*U+Exp(-T)*PY;  {Calcurate : filtering as LPF       }
   WRITELN(Y:7:3);
   DELAY(100);
   PY:=Y;
UNTIL KEYPRESSED;
END.
```

환은 결국 시간영역에서 주파수영역으로 변환하는 것이고 역변환은 그 반대의 변환을 의미한다.

이때 주파수 영역에서는 주파수와 위상이라는 두가지 성분으로 구성된다. 필터링을 함은 주파수에 있어서 필요한 대역과 불필요한 대역을 선별하여 가려주는 것이기 때문에 주파수 영역의 스펙트럼을 직접적으로 손질하여 이를 시간영역으로 역변환하면 원하는 형태의 필터링을 시킬 수 있게 된다.

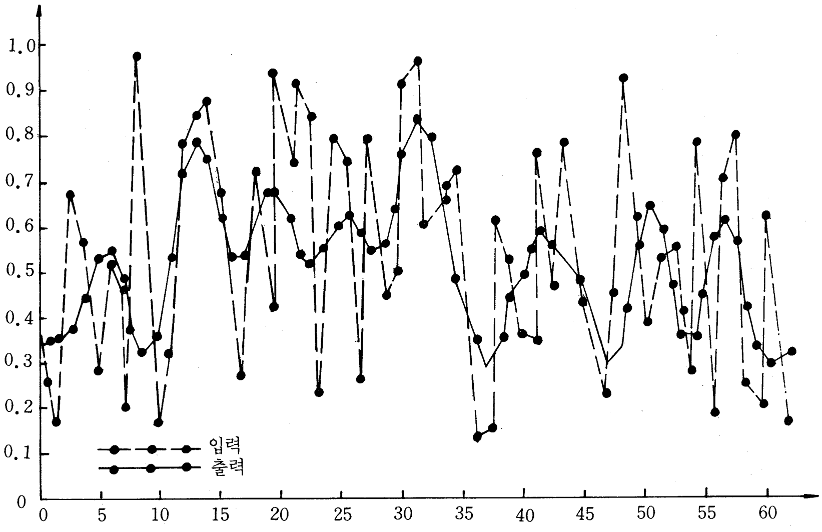
그러면 위의 블록도를 실제로 컴퓨터를 사용하여 구현하기 위한 방법으로 **리스트 2**를 보자. 여기서는 필터의 효과를 보기위해 임의의 신호를 의사난수로써 발생시켜 이를 저역필터에 걸어 필터링한 결과를 보였다.

그림 6은 리스트 2에 대한 결과를 수치로 나타낸 것이며 입력신호보다 훨씬 변동이 적은 결과를 보이고 있다.

여기서 잠시 고등학교때 배운 구분구적법에 대해 생각해 볼때 거기에서는 샘플링수를 늘리면 늘릴수록 적분결과가 정확해진다고 하였다. 그러나 샘플링수 N을 늘린다고 해서 필터링이 더욱 잘되는 것은 아니다.

이것에 관하여는 샘플링이론을 알아야 하는데 간단히 그 개요만 설명하겠다. 어떤 주파수 성분을 갖는 신호를 샘플링하려면 그 주파수 2배이상의 샘플링주파수를 사용하여야 한다. 만일 그렇지 않으면 에일리어싱에러가 발생하고 이는 원래의 신호를 원래의 모습대로 복원할 수 없음을 의미한다. 즉 충분한 샘플링주파수를 사용한다면 샘플링수를 늘려도 실제의 필터링효과는 크게 개선되지는 않는다.

〔그림 6〕 FFT를 통한 디지탈필터의 결과치



(2) 리니어 커브피팅

일반적으로 일군의 데이터를 분석하는 경우 일련의 데이터열의 앞뒤 상관관계를 이용하여 그것의 관계를 도출하여 함수로 표시하는 것을 커브피팅(Curve Fitting)이라고 한다.

이것은 그 데이터의 시간진행과 데이터의 변환곡선이 어떤 형태인가를 고려하는 것이다.

그러면 왜 이러한 방법이 필요한가를 생각해 보자. 예를 들어 국민학교의 과학시간에 암상자를 가지고 입력과 출력을 관찰하여 그것의 내부를 유도해 내는 실험을 기억해 보자. 이것은 우리가 어떤 신호를 받아 그것의 규칙을 발견한다면 수식적으로 기술할 수 있게 되어 쉽게 그 신호의 성질을 알 수 있다는 것이며, 다시 말해 신호를 바탕으로 그것을 모델링할 수 있는 것이다.

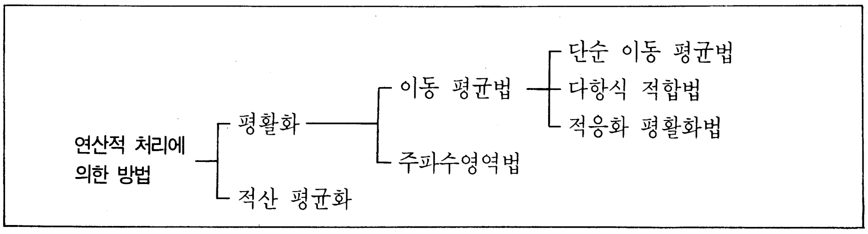
일반적으로 이러한 모델링을 하는 경우에는 신호의 형태를 대강 예상하고 표현가능한 일반식의 계수를 구하는 문제로 귀착한다. 따라서 이런 모델링을 하였을 경우 앞으로의 데이터를 예측할 수 있다.

다음의 **리스트 3**이 실용하면 구해지는 계수의 값이 외부의 외란 즉 노이

즈에도 불구하고 실제의 계수와 근사함을 알 수 있을 것이다. 그러나 이러한 노이즈가 커지면 커질수록 구해진 계수는 부정확해질 것이다.

(3) 그밖의 필터 구현방법

앞에서 설명한 디지탈필터를 실현할 수 있는 방법은 이밖에도 여러가지가 있는데 그중에서 몇가지 방법을 다음에 나타낸다. 사실 다음의 방법들은 시그날프로세싱이란 분야에서 많이 사용되고 있는 것들이다. 특히 영상처리의 경우 평활화에 의한 방법으로 처리하는 경우가 많다. 이외의 다른 방법들은 알고리즘을 어떻게 하느냐에 따라 다르다.



〔리스트 2〕 FFT를 통한 디지탈 필터

```
PROGRAM DIGITAL_FILTER_VIA_FFT;

USES CRT;
TYPE DIM = ARRAY[1..64] OF REAL;
VAR I    :INTEGER; NN:REAL;
    XR,XI:DIM;
    HR,HI:DIM;

PROCEDURE FFT(FLAG,N,IR:INTEGER; DELTA:REAL; Var RX,IX:DIM);
Label  LOOP1,LOOP2,Next;
VAR    i,ir1,L,K,K1,KPL,IS  : INTEGER;
       ARG,Pn,RX1,IX1,NN,m     : REAL;

FUNCTION BitRev(K,IR:INTEGER):INTEGER;
Var K1,K2,BIT,I:INTEGER;
BEGIN
 BIT:=0;
 K1:=K;
 For I:=1 to IR DO
               BEGIN
                K2:=K1 DIV 2;
                 BIT:=2*BIT+(K1-2*K2);
                K1:=K2
               END;
 BitRev:=BIT;
END;

FUNCTION PWR(RADIX,POWER:INTEGER):INTEGER;
BEGIN
  PWR:=Round(Exp(Power*Ln(Radix)));
END;

BEGIN
K1:=0; Pn:=2*pi/N; ir1:=ir-1; L:=N div 2;
FOR IS:=1 TO IR DO BEGIN
LOOP1:For i:=1 to L DO
BEGIN
 K:=K1+1;
         KPL:=K+L;
         m:=BitRev(K1 div pwr(2,ir1),ir);
         IF m=0 THEN BEGIN
                      RX1:=RX[KPL];
                      IX1:=IX[KPL];
                      Goto LOOP2;
                     END;
         ARG:=m*Pn;
         RX1:=Cos(ARG)*RX[KPL]+FLAG*Sin(ARG)*IX[KPL];
         IX1:=Cos(ARG)*IX[KPL]-FLAG*Sin(ARG)*RX[KPL];
LOOP2:RX[KPL]:=RX[K]-RX1;
        IX[KPL]:=IX[K]-IX1;
         RX[K]:=RX[K]+RX1;
         IX[K]:=IX[K]+IX1;
         K1:=K1+1;
        END;{END of i}
 K1:=K1+L;
 If K1 < N then Goto LOOP1
           else BEGIN
                 K1:=0;
                 ir1:=ir1-1;
                 L:=L div 2;
                END;
END;{END of IS}

For K:=1 to N DO
BEGIN
K1:=BitRev(K-1,ir)+1;
IF K1<=K THEN Goto Next
         ELSE BEGIN
               RX1:=RX[K];
               IX1:=IX[K];
                RX[K]:=RX[K1];
                IX[K]:=IX[K1];
                RX[K1]:=RX1;
                IX[K1]:=IX1;
               END;
Next:END;{END of K}
```

```
IF delta<>1 THEN BEGIN
           For K:=1 to N DO
           BEGIN
            RX[K]:=delta*RX[K];
            IX[K]:=delta*IX[K];
           END;
          END;
END;{OF PROCEDURE}


(***** MAIN PROGRAM SECTION **********************************)

BEGIN
CLRSCR;
{  X : INPUT SIGNAL              XR=Real(X) XI=Imag(X)
   H : FILTER CHARACTERISTICS  HR=Real(H) HI=Imag(H)

  FFT PROCESS(FLAG,N,IR,DELTA,XR,XI)
     IF FLAG = +1 THEN FFT
     IF FLAG = -1 THEN Inverse FFT

     N  : SAMPLING DATA NUMBER
     IR : THIS MUST BE POWER OF 2 OF N
          for intance N=64 <-- 2^6
     DELTA : CASE FFT  : TIME INTERVAL
             CASE IFFT : MUST BE 1 (for DIRECT CALCURATION)



(**** Data in x(k) ****)

For i:=1 to 64 DO
            BEGIN
              XR[I]:=5+RANDOM(10); XI[I]:=0;
            END;

 (**** FFT PROCESS ****)

FFT(1,64,6,0.1,XR,XI);
NN:=0.1*64;
 FOR I:=1 TO 64 DO
BEGIN
 XR[I]:=XR[I]/NN; XI[I]:=XI[I]/NN;
 WRITELN(I-1:2,'  ',XR[I],'  ',XI[I],'

       ',Sqrt(Sqr(XR[I])+Sqr(XI[I])));
END;
WRITELN;

(**** SPECTRUM SHAPING ****)

FOR I:=1 TO 11 DO
              BEGIN
               HR[I]:=1; HI[I]:=0;
              END;
FOR I:=12 TO 54 DO
               BEGIN
                HR[I]:=0; HI[I]:=0;
               END;
FOR I:=55 TO 64 DO
               BEGIN
                HR[I]:=1; HI[I]:=0;
               END;

FOR I:=1 TO 64 DO
              BEGIN
               XR[I]:=XR[I]*HR[I]-XI[I]*HI[I];
               XI[I]:=XR[I]*HI[I]+XI[I]*HR[I];
              END;

(**** Inverse FFT PROCESS ****)

FFT(-1,64,6,1,XR,XI);
FOR I:=1 TO 64 DO
BEGIN
 WRITELN(I-1:2,'  ',XR[I]:7:3,'  ',XI[I]:7:3,'

          ',Sqrt(Sqr(XR[I])+Sqr(XI[I])):7:3);
END;
WRITELN;
END.
```

끝으로

이제까지 ADC를 통하여 입력된 신호의 처리에 관하여 미흡하나마 간단히 설명해 보았다. 앞에서는 데이터의 처리에 관하여 비교적 여러가지 방법을 보였는데 실제 사용할 때는 경우에 따라 적합한 방법을 선택을 해야할 것이다. 예제 프로그램의 경우는 파스칼을 사용하였으나 실제의 데이터의 실시간처리를 위해서는 기계어를 사용하는 것이 바람직하다. 하지만 입력데이터가 주기성을 가질때는 기계어를 사용하지 않아도 될 것이다.

결국 입력되는 신호의 종류에 따라 적합한 알고리즘과 언어를 선정하는 것이 핵심이 된다.

그러기 때문에 필자는 각자 나름대로의 독자적인 알고리즘을 작성하여 보다 간단하고 정확하게 데이터를 처리할 수 있는 방법이 나타나기를 기대한다.

〔리스트 3〕 리니어 커브피팅

```
PROGRAM LINEAR_REGRESSION;

TYPE DATA = ARRAY[1..100] OF REAL;
VAR  I,N:INTEGER;
     X,Y:DATA;
     A,B,NOISE,J,K,L,M,T:REAL;

BEGIN
WRITE('- INPUT THE DATA POINTS NUMBER

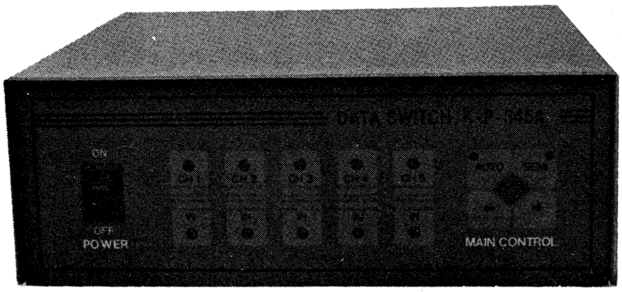
            (LESS THEN 100) : '); ReadLn(N);
FOR I:=1 TO N DO
BEGIN
 NOISE:=Random(3);
 X[I]:=I;
 Y[I]:=2*X[I]+5+NOISE;  {REAL A IS 2 REAL B IS 5}
END;
```

```
J:=0; K:=0; L:=0; M:=0; T:=0;
FOR I:=1 TO N DO
BEGIN
 J:=J+X[I];
 K:=K+Y[I];
 L:=L+Sqr(X[I]);
 M:=M+Sqr(Y[I]);
 T:=T+X[I]*Y[I];
END;
A:=(N*T-K*J)/(N*L-Sqr(J));
B:=(K-A*J)/N;
WRITELN('Y(X) =',A:7:3,'X +',B:7:3);
END.
```

PRINTER 주변기기 전문 메이커

- 컴퓨터는 2대 혹은 3대이고, 프린터는 1대뿐이라 불편하게 사용하고 계십니까?
- 컴퓨터 1대에 프린터를 1대씩 사용하거나 추가 구입시 프린터를 구매하려고 생각하십니까?
- 프린터를 하고 있는 동안 컴퓨터를 장시간 사용하지 못하고 계십니까?
- CAD 사용시 Plotter가 그리고 있는 동안 컴퓨터를 장시간 사용하지 못하고 계십니까?
- 컴퓨터의 수량과 프린터의 수량이 안맞아 적절한 사용을 못하고 계십니까?
- 학교 및 학원의 실습실에 프린터대수가 부족해 적절한 교육을 못하고 계십니까?
- F.A 혹은 계측기쪽의 프린터 인터페이스에 문제가 있읍니까?

컴퓨터 사용중에 불편한 위의 모든 문제를 자신있게 처리해 드리므로, 귀사의 업무에 도움이 되어 드리고 싶읍니다!

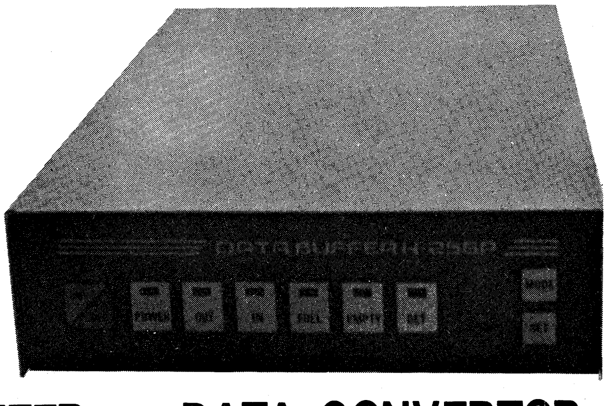


K-P-345A의 사양

입·출력 방식	Centronics Type
입력 COMPUTER	3 : 1~5 : 1
출력 PRINTER	1 대
대기 시간	5초, 10초, 20초, 40초 중 선택
채널 변환방식	전자식
채널 선택방식	자동 or 반자동
용지 F.F 기능	자동 F.F or None F.F
F.F 삭제 기능	자동 삭제 or None

K-256P의 사양

버퍼 용량	256kB(80col 약 200장)
입·출력 방식	Centronics Type
입·출력 수	1 : 1 or 2 : 1
사용자 MODE	10EA
고유 기능	Data 중 Space 입력 기능, 재 Copy 기능, 메모리 기능, Self Test 기능 등 총 10가지



DATA SWITCH

- 전자식 완전자동 (2 : 1~8 : 1 or 64 : 1)
- 전자식 반자동 (2 : 1~8 : 1, 4 : 2)
- Serial 완전자동(4 : 1, 8 : 1)

DATA BUFFER

- 소용량 64kB
- 고성능 다기능 256kB
- 대용량 다기능 1MB
- Serial용 1MB

DATA CONVERTOR

- Serial→PARALLEL
- PARALLEL→Serial

각급 학교 및 학원의 전산 실습실 프린터 공유장치 공사

ATTA SYSTEM
PRINTER 주변기기 전문메이커
서울·종로구 종로 3가 175-4(현대APT709호)
전 화 : 2 7 7 - 1 7 3 9

● 상기 제품은 아래의 전문 취급점에서 전시 판매하고 있으며 직접 사용해 보실 수 있읍니다.●

서울 : 세운상가 협진 274—5419(가동 라열 426호)
용산 나진상가 동해전자 719—8010(17동 가열 304호)
용산 전자랜드 협진 712—2764(3층 C-336호)

인천 : 바이너리 컴퓨터 (032)762—5544
대구 : 범진 컴퓨터 (053)252—2211
마산 : 한길 컴퓨터 (0551)94—5555

(기타 지역 전문 취급점 모집중)

논리함수의 간소화

김정선/항공대 전자과 교수

부울 대수와 논리 게이트

부울 대수(Boolean Algebra)의 기초

1854년에 영국의 수학자 Geroge Boole이 제안한 부울대수는 논리의 수학적 해석을 위한 것으로 논리학 및 기초수학에 응용되었다. 그후 1938년에 미국의 Claude Shannon이 스위치회로의 해석에 응용한 후, 컴퓨터, 전자교환기, 디지탈제어장치 등에 광범위하게 이용되어, 이들의 논리설계와 회로의 특성해석이나 합성에서 실제의 접점이나 소자수의 절약에 성과를 올리고 있다.

부울대수는 0 혹은 1의 값만을 취하는 2진변수와 논리연산자로 구성되며, 이때의 1과 0은 숫자로서의 양을 표시하는 것이 아니고 논리치(참, 거짓)을 표시하는 기호의 의미밖에 없다.

논리연산자로서는 부정(NOT)과 논리합(OR)과 논리곱(AND) 등이 있는데 부정의 기능은 0 혹은 1, 1 혹은 0의 반전된 값을 갖게 된다(그림 1). 논리합은 입력중 하나 이상의 1이 존재할 경우 1의 값을 갖게 되고(그림 2), 논리곱은 모든 입력이 1이어야만 1의 값을 갖게 된다(그림 3). 이들 논리연산자는 집합이론, 벤 다이어그램, 스위치회로에서 쉽게 나타낼 수 있다.

부울 대수의 정리

부울대수에서의 2진변수는 0 혹은 1의 값만을 취하게 되며, 논리연산자에 대해서 교환법칙, 결합법칙, 배분법칙 등이 성립하고, 아래의 정리식들이 성립한다.

① $A+A=A$ $A \cdot A=A$
② $A+0=A$ $A \cdot 1=A$
③ $A+1=1$ $A \cdot 0=0$
④ $A+\overline{A}=1$ $A \cdot \overline{A}=0$
⑤ $A+A \cdot B=A$ $A \cdot (A+B)=A$
⑥ $A+\overline{A} \cdot B=A+B$ $A \cdot (\overline{A}+B)=A \cdot B$
⑦ $\overline{(A+B)}=\overline{A} \cdot \overline{B}$ $\overline{(A \cdot B)}=\overline{A} \cdot \overline{B}$

부울대수중에서 정리 7은 드모르간 법칙으로서 일반화된 표현은

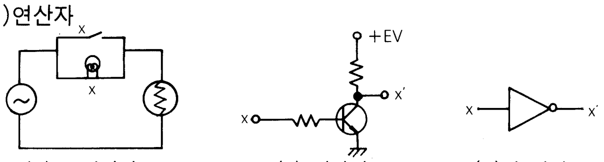
$$\overline{(X_1+X_2+X_3+\cdots+X_n)}=\overline{X}_1 \cdot \overline{X}_2 \cdot \overline{X}_3 \cdot \cdots \overline{X}_n$$

$$\overline{(X_1 \cdot X_2 \cdot X_3 \cdot \cdot X_n)}=\overline{X}_1+\overline{X}_2+\overline{X}_3+\cdots+\overline{X}_n$$

으로 쓸 수 있다. 이 법칙은 0과 1 그리고 +와 ·을 동시에 교환하면 논리식이 성립되는 쌍대(Duality)원리에서

〔그림 1〕 부정(NOT)연산자

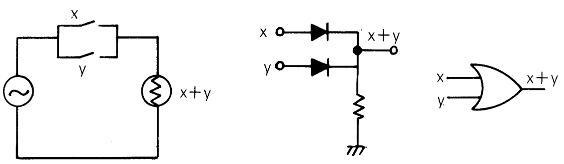
x	x'
0	1
1	0



(a) 진리표 (b) 스위치회로 (c) 전자회로 (d) 논리회로

〔그림 2〕 논리합(OR)게이트

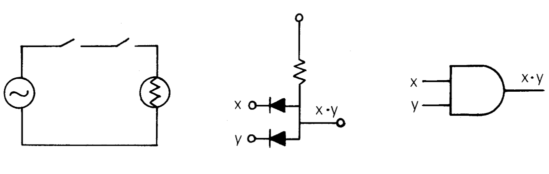
x y	x + y
0 0	0
0 1	1
1 0	1
1 1	1



(a) 진리표 (b) 스위치회로 (c) 전자회로 (d) 논리회로

〔그림 3〕 논리곱(AND)연산자

x y	x · y
0 0	0
0 1	0
1 0	0
1 1	1



(a) 진리표 (b) 스위치회로 (c) 전자회로 (d) 논리회로

유도된다(**그림 4**). 이 법칙은 논리합과 논리곱이 완전히 독립되어 성립되는 것이 아니라 부정을 조합시켜 상호교환이 가능하도록 하는 중요한 정리로서 논리식의 간단화 또는 변환 등에 많이 사용되는 중요한 정리이다.

예를 들어 보면

① $\overline{\overline{x}+\overline{y}}=\overline{\overline{x}}\cdot\overline{\overline{y}}=x\cdot y$
② $\overline{\overline{x}\cdot\overline{y}}=\overline{\overline{x}}+\overline{\overline{y}}=x+y$

위와 같이 부울정리들은 논리회로의 수학적 표현과 복잡한 논리회로의 해석에 사용된다.

예를 하나 더 들어 보자. $z=x(x'+y)$와 같은 논리회로를 구성한다면 **그림 5**(a)와 같은 논리회로로 구현시킬 수 있으나 이것을 부울대수로 간략화시키면 그림 5(b)의 논리회로로 구현시킬 수 있다.

$z=x\cdot(x'+y)=x\cdot x'+xy=xy$

논리 게이트

그림1~3에서 설명한 NOT, OR, AND 게이트외에도 NAND, NOR, 배타적 OR게이트 등이 있다.

NAND게이트는 AND와 NOT를 결합한 게이트이고, NOR게이트는 OR와 NOT을 결합한 게이트이다. 배타적 OR 게이트는 두개의 입력이 서로 같지 않을 때만 출력이 1이 되는 논리회로이다(**그림 6**).

실제의 디지탈회로에서는 NAND나 NOR게이트가 AND나 OR게이트 보다 많이 사용된다. 그 이유로는 NAND 나 NOR게이트는 제조하기가 쉽고 모든 IC의 기본게이트를 구성할 수 있기 때문이다.

NAND게이트와 NOR게이트의 등가회로는 드모르간법칙을 이용하여 쉽게 구할 수 있다(**그림 7**).

NAND게이트와 NOR게이트만으로 구성되는 논리회로를 실현시켜 보자. NAND게이트만으로 부울함수의 논리회로를 구성하려면, 부울함수를 곱항의 합(Sum of Minterms)의 형태로 간소화하여야 한다.

$$W=XYZ+X\overline{Y}Z+\overline{X}\overline{Y}$$

의 함수를 NAND게이트로 논리를 표시하면

$$W=\overline{(XYZ+X\overline{Y}Z+\overline{X}\overline{Y})}$$
$$=\overline{(XYZ)}\cdot\overline{(X\overline{Y}Z)}\cdot(\overline{\overline{X}\overline{Y}})$$

의 식으로 나타나며 **그림 8**과 같다. 그림 8(a)는 AND와 OR회로로 표시한 그림이고 (b)는 NAND게이트로 구성된 그림이다.

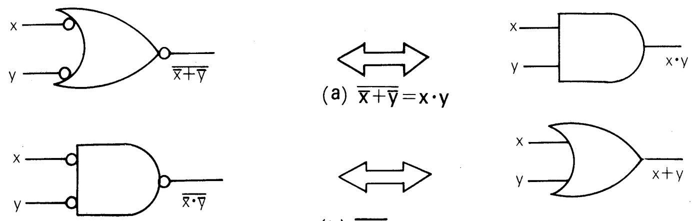
한편 NOR게이트만으로 부울함수의 논리회로를 구성하려면 부울함수를 합항의 곱인 형태로 바꾸고 실현시킬 수 있다. 함수

$$W=\overline{X}Y\overline{Z}+XZ+\overline{Y}$$

를 NOR게이트 형식으로 표시하려면 우선 부울함수를 합항의 곱형태로 바꾸어야 한다(**그림 9**).
그림 9(a)의 진리표에서

$$W=(X+\overline{Y}+\overline{Z})\cdot(\overline{X}+\overline{Y}+Z)$$

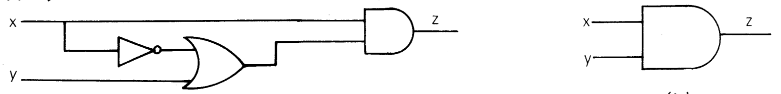
〔그림 4〕 드모르간법칙 사용 예



(a) $\overline{\overline{x}+\overline{y}}=x\cdot y$

(b) $\overline{\overline{x}\cdot\overline{y}}=x+y$

〔그림 5〕 논리회로 간략화의 예 $x\cdot(x'+y)=x\cdot y$



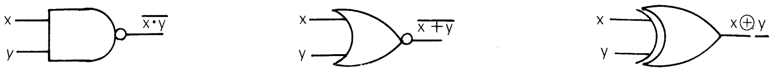
(a) (b)

〔그림 6〕 NAND, NOR, 배타적OR 게이트

x y	$\overline{x\cdot y}$
0 0	1
0 1	1
1 0	1
1 1	0

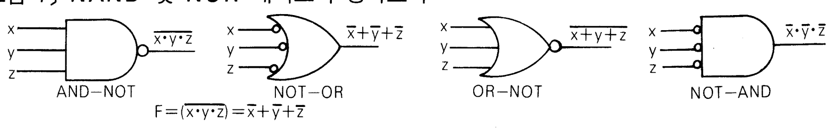
x y	$\overline{x+y}$
0 0	1
0 1	0
1 0	0
1 1	0

x y	x y
0 0	0
0 1	1
1 0	1
1 1	0



(a) NAND (b) NOR (c) 배타적-OR

〔그림 7〕 NAND 및 NOR 게이트의 등가표시



(a) NAND 등가회로 (b) NOR 등가회로

의 합항의 곱을 구할 수 있고, 이 식은 드모르간법칙을 이용하여

$$W = \overline{\overline{(X+\bar{Y}+\bar{Z})\cdot(\bar{X}+\bar{Y}+Z)}}$$
$$= \overline{(X+\bar{Y}+\bar{Z})} + \overline{(\bar{X}+\bar{Y}+Z)}$$

로 얻어지므로 그림 9(c)에서와 같이 NOR게이트로 구성된 논리회로를 얻을 수 있다.

적의합과 합의적

논리표현식을 만들어내는 직접적인 방법은 진리표에서 출력이 1인 항을 읽어내는 것이다. 이 방법은 적의 합(Sum of Products)형식으로 논리식을 표현하는 것이다. 표 1의 진리표에서 적의 합을 구해보자. 우선 진리표에서 정논리인 항을 읽어내어 이들의 합 형태로 만들면 된다. 즉,

$$F = \bar{A}\bar{B}\bar{C} + \bar{A}BC + A\bar{B}\bar{C} + A\bar{B}C$$

의 표현으로 적의 합으로된 논리식을 얻을 수 있다. 위의 진리표에서 합의 적 형태의 논리식을 구해보면, 우선 진리표에서 부논리의 항을 읽어내어 드모르간법칙을 이용해서 얻을 수 있다.

$$\bar{A}\bar{B}\bar{C} = \bar{A}B\bar{C} = AB\bar{C} = ABC = 0$$

이므로 위의 각항들을 드모르간의 원리를 이용하여

$$(A+B+\bar{C}) = (A+\bar{B}+C) = (\bar{A}+\bar{B}+C) = (\bar{A}+\bar{B}+\bar{C}) = 1$$

을 얻고, 이 각 항들을 곱의 형태로 하여 합의 적인 논리표현식을 얻을 수 있다.

각종 논리전자회로의 비교

논리전자회로를 비교함에 있어서 중요한 특성들을 보면 잡음여유도, 스위칭속도, 소비전력, 그리고 팬인(Fan−In) 과 팬아웃(Fan−Out) 등이 있다. 각종 논리전자회로의 특성비교는 표 2에 나타나 있다.

논리함수의 간소화

논리함수는 카르노맵이라는 도시법을 사용하여 편리하게 간소화시킬 수 있다. 카르노맵은 4개이내의 변수로 된 논리함수를 간소화하는데 편리하지만 4개이상의 변수에서는 맵의 기하학적인 어려움으로 사용을 피하고, Quine−McClusky 법이라는 도표법을 사용하는 것이 편리하다.

카르노맵

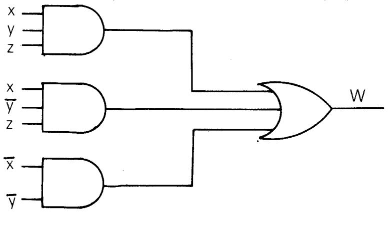
n개의 2진변수 $x_1, x_2 \cdots, x_n$을 갖는 논리함수$f(x_1, x_2 \cdots, x_n)$의 맵은 변수의 가능한 조합인 2^n개의 4각형으로 이루어지고, 이 4각형 안에 논리함수의 값을 기록한다. 따라서 이 4각형 안의 논리함수 값은 1, 0, 혹은 사용되지 않은 논리함수의 4각형에 d의 값으로 표시된다(표 3). 이때 4각형의 배열은 그레이코드에서와 같이 배열되어 각 논리함수 조합도 행이나 열로 한칸씩 움직일 때 마다 1비트의 변화만을 일으키고, 역시 최우칸과 최좌칸, 그리고 최상칸과 최하칸의 변화도 1비트이다.

이러한 카르노맵의 기본원리는 부울

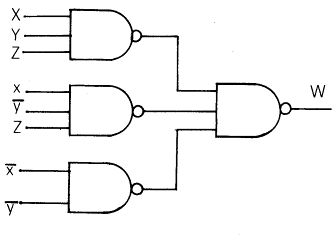
〔표 1〕 진리표

A	B	C	F
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

〔그림 8〕 논리식의 NAND 게이트 구성



(a)

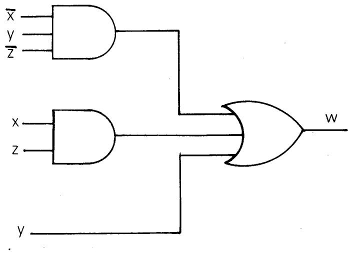


(b)

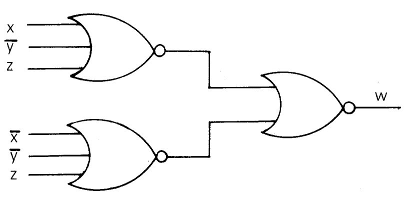
〔그림 9〕 NOR게이트로 구성된 논리회로 예 $W = \bar{X}Y\bar{Z} + XZ + \bar{Y}$

X	Y	Z	W
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

(a) 진리표



(b) AND와 OR로서 구성



(c) NOR로 구성

정리의 흡수법칙

$f(x_1, x_2)=\bar{x}_1x_2+x_1x_2=(\bar{x}_1+x_1)x_2=x_2$

을 이용한 것이다.

$F(A,B,C)=\bar{A}\bar{B}\bar{C}+\bar{A}\bar{B}C+A\bar{B}\bar{C}+A\bar{B}C$
$+ABC$

의 논리함수를 예로서 카르노 맵을 살펴보면, 그림 10(b)의 카르노맵을 구한 후에 정논리의 블록을 형성시켜 간략화 시킬 수 있다.

그림 10(b)에서 정논리 4개의 블록에서 행과 열의 변화가 없는 변수 즉 $\bar{B}$를 얻을 수 있고, 정논리 2개의 블록에서 행과 열의 변화가 없는 변수 AC를 얻고 이의 합 형태로서

$F=\bar{B}+AC$

의 간소화된 최소의 논리함수를 얻을 수 있다. 카르노맵에서의 don't care 0 항은 정논리의 블록을 형성하는 데 도움이 되면 같이 묶어 주어서 최소의 논리함수를 얻을 수 있다.

Quine-McClusky 방법

Quine-McClusky방법은 카르노맵에서 4변수의 난해한 점을 극복하여 4변수 이상의 함수를 간소화시키는 효과적인 방법이다. 이 방법의 기본원리는 진리표로부터의 정논리 식중에서 1의 갯수가 같은 항끼리 모아 표를 만든 후, 변수가 하나씩 변환하는 것을

$xy+x\bar{y}=x(y+\bar{y})=x$

를 이용하여 최소의 논리식인 주항(Prime Implicants)을 구하는 방법이다. 표 4의 예를 통해 살펴보자.

진리표에서 1의 값을 갖는 변수의 갯수가 같은 항끼리 모아서 배열표를 만들고, 배열의 첫항으로부터 1의 값을 갖는 변수의 갯수가 하나 더 많은 배열의 각 항들을 비교하여 동일열상의 한 변수만 변화한 항은 각각 V표시를 하고 변화된 비트자리에는 -를 채워서 제1감축표에 모아두고, 위의 과정을 반복한다. 이때 감축에 참여하지 못한 항은 0표를 한다. 즉, $xy+x\bar{y}=x$에 의해 감축된 항은 감축표를 구성하게 되고 감축에 참여하지 못한 항은 0표를 하여 선별표에서 진성주항(Essential Prime Implicant)를 찾기 위함이다.

제1감축표가 완성되면 앞의 과정들을 반복하여 제2감축표를 만들고 이와 같은 과정을 감축이 일어나지 않을 때까지 계속한다. 배열표와 감축표에서 0표가 되어 있는 항은 감축에 참여하

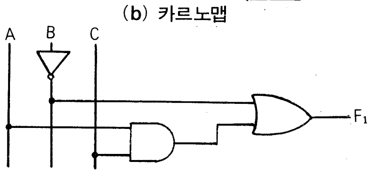
〔그림 10〕 $F(A, B, C)=\bar{A}\bar{B}\bar{C}+\bar{A}\bar{B}C+A\bar{B}\bar{C}+A\bar{B}C+ABC$의 간략화

A	B	C	F_1
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

(a) 진리표

A \ BC	00	01	11	10
0	1	1	0	0
1	1	1	1	0

(b) 카르노맵



(c) 논리회로

〔표 2〕 논리회로의 비교

항목 \ 논리회로형식	TTL	CMOS	ECL (MECLⅢ)	HTL	DTL	RTL
기본회로 (fan out)	NAND 10	NAND 또는 NOR 50	OR/NOR 7	NAND 10	NAND 8	NOR 5
소비전력 (게이트당)	1〔mW〕(74L) 10〔mW〕(74) 19〔mW〕(74S)	15〔μW〕	240〔mW〕	55〔mW〕	8~12〔mW〕	12〔mW〕
잡음여유	0.4〔V〕	2.4〔V〕	0.3〔V〕	—	—	—
전달지연시간 (게이트당)	33〔ns〕(74L) 10〔ns〕(74) 3〔ns〕(74S)	25〔ns〕	0.5〔n〕	150〔ns〕	30〔ns〕	12〔ns〕
공급전원전압	5〔V〕	3~18〔V〕	−5.2〔V〕	15〔V〕	5〔V〕	3〔V〕 3.6〔V〕

〔표 3〕 2변수, 3변수, 4변수일 때의 배열

f_1

x_1 \ x_2	0	1
0	f(0, 0)	f(0, 1)
1	f(1, 0)	f(1, 1)

(a) 2변수

f_2

x_1 \ x_2x_3	0 0	0 1	1 1	1 0
0	f(0, 0, 0)	f(0, 0, 1)	f(0, 1, 1)	f(0, 1, 0)
1	f(1, 0, 0)	f(1, 0, 1)	f(1, 1, 1)	f(1, 1, 0)

(b) 3변수

f_3

x_1x_2 \ x_3x_4	0 0	0 1	1 1	1 0
0 0	f(0, 0, 0, 0)	f(0, 0, 0, 1)	f(0, 0, 1, 1)	f(0, 0, 1, 0)
0 1	f(0, 1, 0, 0)	f(0, 1, 0, 1)	f(0, 1, 1, 1)	f(0, 1, 1, 0)
1 1	f(1, 1, 0, 0)	f(1, 1, 0, 1)	f(1, 1, 1, 1)	f(1, 1, 1, 0)
1 0	f(1, 0, 0, 0)	f(1, 0, 0, 1)	f(1, 0, 1, 1)	f(1, 0, 1, 0)

(c) 4변수 때의 도시

〔표 4〕 Quine-McClusky법

A B C D	F
0 0 0 0	1
0 0 0 1	0
0 0 1 0	0
0 0 1 1	0
0 1 0 0	1
0 1 0 1	1
0 1 1 0	1
0 1 1 1	0
1 0 0 0	0
1 0 0 1	0
1 0 1 0	0
1 0 1 1	1
1 1 0 0	1
1 1 0 1	1
1 1 1 0	1
1 1 1 1	1

(a) 진리표

항배열		제 1 감축		제 2 감축	
A B C D	검표	A B C D	검표	A B C D	검표
0 0 0 0	∨	0 — 0 0	○	— 1 0 —	○
0 1 0 0	∨	0 1 0 —	∨	— 1 — 0	○
0 1 0 1	∨	— 1 0 0	∨	1 1 — —	○
0 1 1 0	∨	0 1 — 0	∨		
1 1 0 0	∨	— 1 0 1	∨		
1 0 1 1	∨	— 1 1 0	∨		
1 1 0 1	∨	1 1 0 —	∨		
1 1 1 0	∨	1 1 — 0	∨		
1 1 1 1	∨	1 — 1 1	○		
		1 1 — 1	∨		
		1 1 1 —	∨		

(b) 감축표

〔표 5〕 선별표

주항 A B C D \ 최소항	0000	0100	0101	0110	1011	1100	1101	1110	1111
0 — 0 0	⊗	×							
— 1 — 0		×		⊗		×		×	
1 — 1 1					⊗				×
— 1 0 —		×	⊗			×	×		
1 1 — —						×	×	×	×

지 못한 주항이 되는 것이다.

다시 이 주항들을 효과적으로 축소 결합시키기 위하여 표 5와 같이 가로축에 정논리의 논리식을 배열하고 세로축에는 위에서 얻어진 주항들을 배열한다. 세로축의 각 주항의 —표에 0과 1을 조합하여 가로축과 관련된 교점에 ×표를 한다. 즉 감축되기 전의 항이 감축되어 주항으로 된 관계를 표시한 것이다.

표 5의 선별표에서 세로로 교차점이 하나 밖에 없는 주항은 진성주항이 되고 각 진성주항의 교차점들과 동일열상에 있는 교차점을 가진 주항들은 각 진성주항으로 감축된다. 앞의 선별표에서 1 1 — — 주항이 이에 해당된다. 따라서 표 4(a)의 진리표를 갖는 논리함수는

$$F=\bar{A}\bar{C}\bar{D}+B\bar{D}+ACD+B\bar{C}$$

로 감소되어 얻어질 수 있다. PC

하드디스크의 설치

컴퓨터를 사용할 때 하드디스크가 없이 플로피디스크만을 사용한다면 여러 유틸리티 프로그램을 이용하거나 데이터를 세이브나 로드하기가 매우 불편하다.

사실상 하드드라이브는 여러가지 응용 소프트웨어를 사용할 때 많이 이용되며 그것은 마이크로프로세서 만큼이나 중요하다고 말할 수 있다. 대부분 사용자들은 하드디스크를 구입하려면 많은 돈이 필요하다고 생각하고 있고 돈을 갖고 있다 하더라도 그 설치과정이 꽤 어렵다고 느끼고 있기 때문에 하드디스크의 설치를 꺼린다. 그러나 다행히도 하드디스크를 PC-XT와 AT에 설치하기는 간단하며 실제로 모든 PC 사용자들은 아무 문제없이 설치작업을 할 수 있다.

설치전 알아야 할 기본 사항들

하드디스크를 설치하고자 할때 가장 먼저 당면하는 문제는 내가 필요로 하는 것이 무엇인가? 하는 점이다. 대부분의 PC는 2대의 하프사이즈 FDD와 2대의 하프사이즈 HDD 등 4대를 동시에 내장할 수 있게 되어 있다. 풀사이즈 드라이브를 내장한다면 FDD를 2대 설치하거나 FDD 1대, HDD 1대 등 총 2대를 설치할 수 있다. 일부분의 기종은 더 많은 드라이브를 내장할 수 있게 케이스가 변형되어 있으나 대부분의 경우는 앞의 경우와 동일하다.

AT시스템은 대체로 두가지 드라이브 형태를 제공할 수 있는 결합된 하드/플로피 디스크 제어보드를 사용하며 반면 XT시스템은 하드디스크 제어보드와 플로피디스크 제어보드를 따로 사용한다. 만일 전원 연결자를 사용할 수 없다면 플로피디스크 드라이브와 같은 다른 장치로 가는 분리된 전원케이블인 T케이블(회선)을 청계천 전자부품상가등에서 구입해야 한다.

하드디스크의 설치전에 먼저 확인해야 하는 사실은 파워서플라이의 용량이다. AT의 경우 230W, XT는 150W 정도의 정격전류가 되어야 하며, 시스템에 추가로 보드를 설치하거나 할 때에는 추가장치를 지원할 수 있을 만큼 충분한 용량을 가지고 있어야 한다.

만일, 파워서플라이의 용량이 이보다 모자란다면 보드의 추가가 불가능하다. 이러한 사실들을 무시하고 추가장비를 설치한다면 시스템이 제대로 동작하지 않는다.

다음으로 주의할 점은 슬롯에 꽂혀 있는 보드들의 숫자이다. 8개 슬롯 전체에 확장보드가 전부 꽂혀 있다면 HDD 등을 확장할 수가 없다. 일반적으로 시스템에 필요한 확장장치로는 모노크롬용 비디오 보드나 컬러 그래픽용 비디오 보드, 주변기기의 입출력에 필요한 시리얼/패러렐 보드, FDD, HDD 보드, 그리고 각자의 용도에 따라 필요한 램 확장 보드나 CPU 보드, 멀티펑션 보드, 리얼타임클럭 보드 등이다. 따라서 시스템을 제대로 활용하려면 항상 5~7개 정도의 확장 보드가 슬롯에 꽂혀 있게 된다. 확장 보드 설계시에는 이점에 유의하여 확장시 각각 용도에 따른 것을 구입할 것인지 여러가지 용도용의 다기능 보드를 선택할 것인지 등을 결정하여야 한다.

그 다음에는 인스톨시 DOS버전의 선택에 유의해야 한다. 하드 디스크의 용량에 따라 DOS버전에 따른 최대 파티션이 달라지므로 가급적이면 최신 버전의 것을 선택하는 것이 좋다.

AT용 하드디스크와 XT용 하드디스크에 있는 기계적인 장비구조들은 조금 다르다. 즉 AT시스템들은 한쪽 슬라이드(Slide) 구조이고 XT시스템은 그렇지 않다. 그래서 만일 AT에서 XT시스템을 위해고안된 하드디스크를 설치하려고 하면 기계적인 설치가 잘 되지 않는다.

대부분 XT시스템들은 하드디스크 제어보드를 내장하고 있지 않기 때문에 이러한 모델들을 위한 하드디스크 부시스템들은 일반적으로 미리 구성된 제어보드와 결합된 케이블들이 호환성있게 판매된다.

표준 XT와 AT 하드디스크들은 제어기와 드라이브사이에 ST-506/412 인터페이스를 사용한다. 시게이트사에 의해 개발된 이 인터페이스는 ST-506과 ST-412 디스크 드라이브에 처음 사용되었다.

다른 표준인터페이스들로는 향상된 소형 디스크 인터페이스(ESDI)와 소형 컴퓨터시스템 인터페이스(SCSI)가 있으며 이러한 드라이브 시스템들의 설치과정은 비슷하다.

설치작업

하드디스크 설치과정을 설명하기전에 몇가지 주의사항을 말하면 다음과 같다.

우선 컴퓨터 본체안에서 작업할 때 구성소자들의 충격과 손실을 방지하기 위해서AC 전원코드를 뺀다.

두번째로 하드디스크 제어 보드와 그 자체에 MOS회로를 포함하고 있는 하드디스크는 정전기에 민감하다. 설치하기 위해 필요할 때까지 그것들을 건드리지 말고 일단 설계하기 전에 축적된 어느 정전기를 방전하기 위해 접지부분을 건드린다. 이것은 AC 전원코드를 끊기전에 컴퓨터에 있는 파워 서플라이를 건드림으로써 할 수 있다.

마지막으로 비록 잘 알고 있다 하여도 하드디스크는 섬세한 장치들이므로 조심해서 다루어야 한다. 이런 사항들을 숙지한 다음 우선 컴퓨터의 덮개를 열고 드라이브를 넣을 수 있도록 공간을 확보한다. 경우에 따라서는 하드디스크가 들어갈 자리를 만들기 위해 플로피디스크를 옮길 필요가 있을 수도 있다.

만일 디스크 제어보드가 없는 시스템을 가지고 있다면 디스크드라이브 제어 보드를 쓸 수 있도록 여분의 슬롯을 확보해야 한다. 하드디스크는 점퍼블록이나 드라이브 번호를 선택하기 위한 딥스위치를 가지고 있다. 이것은 하드디스크를 만들때 이미 만들어져 있다.

조심스럽게 디스크드라이브를 컴퓨터 앞쪽에서 비어있는 공간으로 가만히 집어 넣는다. 디스크드라이브의 인쇄된 회로기판쪽이 아래에 가도록 한다. 뒤로 너무 깊이 드라이브를 집어넣지 말아라. 회선 연결을 위해 공간이 필요할지 모르기 때문이다. 드라이브는 금속기판위에 놓는다. 만일 플로피디스크같은 다른 드라이브위에 하드디스크를 놓으려면 두 드라이브들 사이에 금속분리기판을 끼우는 것이 좋다.

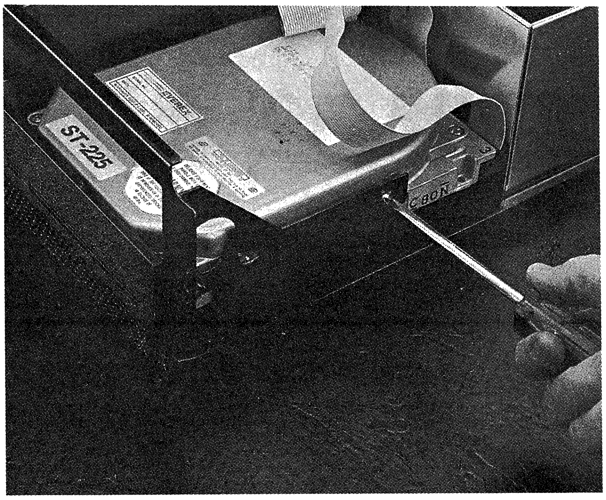
표준 ST-506 인터페이스를 가지고 있는 하드디스크는 20과 34핀 케이블인 두 케이블로 구성되어 있다. 두 회선은 하드디스크와 그것의 제어 보드를 연결한다. 각 케이블들은 줄무늬로써 케이블의 1번 핀을 구분하고 있다. 하드디스크 제어 보드에 해당하는 연결 코넥터에도 또한 1번 핀을 지시하는 표시가 있다. 이것은 보통 점, 화살표, 숫자 1 또는 때때로 핀 근처에 숫자 2로 지시하였거나 "Pin 1."라는 말로 핀에 적어놓는 등 여러 형태로 되어 있다. 대부분의 하드디스크 설치시의 문제점들은 부정확한 케이블링 때문에 일어나므로 케이블의 이중점검이 필요하다.

다음으로 전원공급에서 하드디스크의 전원연결소자까지 4핀 전원 케이블을 연결한다. 전원 연결소자는 쐐기가 있어 단지 한쪽으로 전원 케이블을 삽입할 수 있다. 만일 XT기종을 가지고 있고 하드디스크를 설치하기 위해 플로피디스크 드라이브를 옮겼다면 플로피 드라이브들과 다르게 딥스위치를 재구성해 주어야 한다.

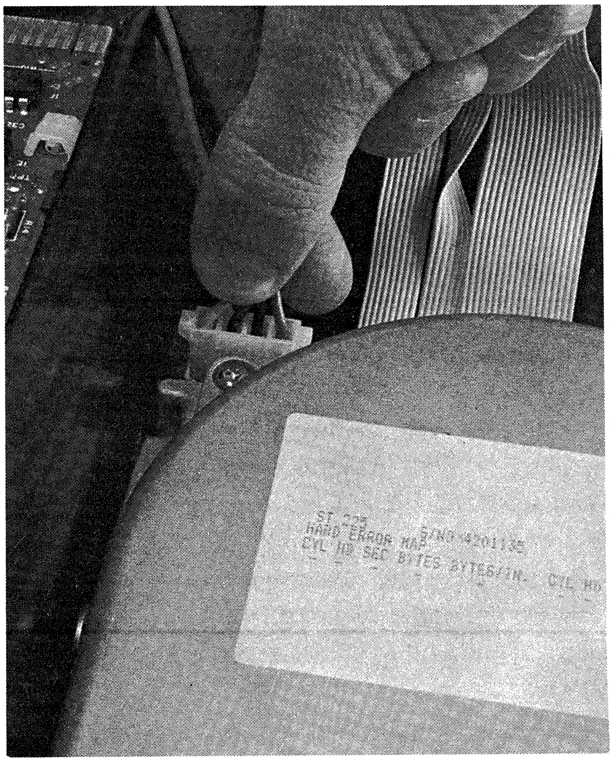
적당한 딥스위치 구성법과 작동법은 반드시 컴퓨터의 매뉴얼을 참고하는 것이 안전하다.

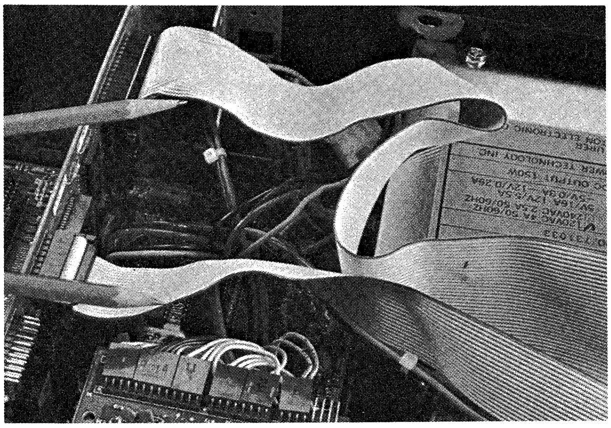
하드디스크가 설치되었다면 다음과 같은 소프트웨어 작업을 해야 한다. 덮개를 제자리에 돌려놓기 전에 드라이브가 적당히 움직이는지 확인하기 위해 AC전원 케이블을 재연결시키고 컴퓨터에 전원을 올려라. 만일 하나나 두개의 플로피디스크 드라이브를 가지고 있다면 새하드디스크는 드라이브 C가 될 것이다. 두 플로피디스크 드라이브 이상을 가지고

하프 사이즈의 HDD를 설치할 때는
대개 FDD 밑쪽이나 시스템 안쪽에 보이지 않도록 설치한다.



4 핀의 전원케이블이 이상 있으면 하드디스크가 작동하지 않는다.





아래쪽에는 20핀용 드라이브 C의 제어신호용 코넥터가 있고 나머지 여분 하나는 드라이브를 추가할 때 사용한다.

있다면 새 하드디스크는 다음으로 이용가능한 논리적 드라이브 이름이 될 것이다. 하드디스크는 그것을 사용하기 전에 3단계의 준비작업을 거쳐야 한다.

처음단계는 포맷팅 또는 초기화단계인데 대부분의 하드디스크들은 이미 이 단계를 거쳐서 온다. 다음 단계는 분할작업이다. 시스템 DOS 디스켓내의 FDISK를 이용하여 도스 파티션을 생성시킨다. 세팅에 관한 자세한 사항은 퍼스널컴퓨터 88년12월호 210페이지 '하드디스크의 셋업'란을 보면 자세히 나와 있다.

만일 하드디스크 용량이 32.5MB 이상이고 DOS 3.1 또는 그 이상의 버전을 사용하지 않는다면 다중 논리적 드라이브들로 드라이브를 분할한다. 가지고 있는 기종이 XT가 아니고 AT 컴퓨터를 가지고 있다면 새로운 하드 디스크에 관한 컴퓨터를 말하는 이 시점에서 Setup(설치) 유틸리티를 실행할 필요가 있을 것이다.

대부분 하드디스크의 설치 문제점들은 부적당하게 연결된 케이블 때문에 일어난다. 다른 잠재적인 문제점들은 충분치 못한 전원공급에서 야기된다. 시스템에 전원을 넣었을 때 별 같은 모양의 것이 화면에 어지럽게 나오게 되면 전원을 끄고 다시 켠다. 전원공급이 너무 과다하기 때문이다. 종종 컴퓨터에 있는 모든 불필요한 기판들을 제거함으로써 때때로 시스템이 부트가 안되는 것을 방지할 수도 있다. 대개 전원공급의 문제 때문에 발생한 일이기 때문이다.

주의사항

만일 컴퓨터를 옮기려고 한다면 하드디스크 헤드는 디스크 플레터들의 중요한 영역으로부터 멀리 옮겨 놓아라. 대부분 AT드라이브들과 몇개의 XT드라이브들은 오토 파크 헤드(Auto Park Head)를 가지고 있다. 그러나 대부분의 XT드라이브들은 오토 파크 헤드를 갖고 있지 않다. 많은 하드디스크 부시스템들은 이런 목적을 위해 파크 프로그램을 포함하고 있는 유틸리티 디스크를 가지고 있다. 만일 이런 유틸리티를 가지고 있지 않다면 시장에 있는 소프트웨어 패키지들에서 이런 기능을 얻을 수 있다.

마지막으로 남기고 싶은 말은 항상 하드디스크 정보를 백업해두란 말이다. 때로 하드디스크 드라이브들이 고장날 수 있고 따라서 정규적인 백업작업은 매우 중요하다.

〈용어해설〉

● **머지(Merge)**
조합이라고 하며 어떤 순서로 레코드가 배열된 복수개의 화일을 대조하여 그 순서에 따라서 하나의 화일에 집약하는 것을 말한다.

● **멀티태스킹(Multitasking)**
복수개의 태스크를 병행, 동작시키는 것을 말한다. 태스크는 CPU, I/O, 주기억장치 등의 각 부위를 점유하여 동작을 한다. 그러나 태스크는 사상대기이기 때문에 일시적으로 조작권이 포기되는 일이 있으며 이 대기 시간을 다른 동작 가능한 태스크에 할당하여 그 태스크를 동작시킬 수가 있다.

● **메모리 사이클(Memory Cycle)**
메모리의 지정된 어드레스에 데이터를 라이트하거나 지정된 어드레스로부터 데이터를 리드하는 일련의 동작으로 이것에 요하는 시간을 메모리 사이클 타임이라고 한다.

● **명령실행 사이클(Instruction Execution Cycle)**
머신 사이클의 하나로써 메모리에서 하나의 명령을 CPU내에 끌어들여서 해독한 후 그 명령의 실행이 종료하기 까지를 말한다.

● **전송(Transfer)**
하나의 기억장소(예를 들면 10번지의 메모리) 또는 레지스터에 있는 정보(명령이나 데이터)를 다른 기억장소(예를 들면 20번지의 메모리) 또는 레지스터에 옮기는 것을 말한다.

● **로드(Load)**
주변장치(카드리더, 디스크 메모리 등)에서 프로그램이나 데이터를 CPU의 내부 메모리에 보내어 이들을 기억시키는 것. 지정한 메모리의 내용을 어큐뮬레이터 등의 레지스터로 옮기는 것, 또는 그 명령을 말한다.

● **서비스 프로그램(Service Program)**
사용자가 프로그램을 만들거나 데이터 처리를 할 때 사용자를 도와주기 위하여 제조회사가 제공하는 프로그램.

애플과 IBM-PC와의 통신

김정현/항공대 대학원 전자과

몇년전 부터 국내에 컴퓨터열풍이 불기 시작한 이래 많은 수의 8비트 컴퓨터들이 보급되었는데 그중 주종을 이룬 것은 가격대 성능비가 뛰어난 애플II였다. 이 애플II는 그런데로 쓸만한 물건이었지만 메모리 용량부족과 한글이식의 실패로 업무용 컴퓨터로 정착되지 못하고 16비트 IBM-PC에 인기를 빼앗겼다. 지금은 IBM-PC XT나 AT 기종들이 널리 보급되고 있다.

그러나 지난 수년동안 애플용으로 수많은 프로그램들이 작성되고 데이터베이스가 만들어 졌는데 그중 상당수는 그대로 혹은 약간의 수정을 거쳐 IBM-PC상에서 수행될 수 있다.

이런 프로그램들은 IBM-PC에서 사용하려면 컴퓨터 통신을 이용하는 수 밖에 없다. 애플II와 IBM-PC는 디스크 포맷이 다르기 때문에 애플II용 디스켓을 IBM-PC가 읽어들일 수 없고 이들을 일일이 키보드로 입력하는 것도 가능은 하지만 너무 힘든 일이기 때문이다.

본고에서는 애플II와 IBM-PC간에 RS-232C 인터페이스를 통해 데이터를 주고받는 방법을 설명하겠다. 우선 컴퓨터 통신기술의 기초적인 사항부터 알아보기로 하자.

직렬통신과 병렬통신

컴퓨터와 컴퓨터, 혹은 컴퓨터와 주변장치 사이의 데이터 전송방법은 가장 크게 이 두가지로 구분된다. 병렬(Parallel)전송은 하나의 워드(이 경우 대개 한 바이트)를 워드를 구성하는 비트수 만큼의 신호선을 통해 동시에 전송하는 방법이다. 컴퓨터내부에서 CPU와 메모리간의 데이터 전송은 병렬전송에 해당된다. 프린터에 많이 이용되는 센트로닉스 인터페이스가 병렬전송 인터페이스의 대표적 예이다. 이 병렬전송방법은 속도가 빠르고 간단하나 접속 케이블의 선이 많아지고 원거리 전송에 부적합하여 대개 1m 이내의 거리에서 이용된다.

직렬(Serial)전송은 한 워드를 비트단위로 분해하여 한 라인의 전송선을 통해 한순간에 한 비트씩 전송하고 수신측에서는 받은 비트들을 한 워드로 재조립하는 방법이다. 이 경우 송신측에서 워드를 비트열로 분해하는 방법과 수신측에서 비트열을 워드로 재구성하는 방법이 동일해야 하며 송신속도(보내지는 비트들의 시간간격)와 수신속도도 반드시 같아야 한다. 송신측과 수신측의 이러한 규약을 프로토콜(Protocol)이라 한다(프로토콜이란 용어의 정확한 뜻은 더 넓은 범위에서 정의된다).

직렬전송은 병렬전송에 비해 훨씬 먼거리 전송에 이용될 수 있다. 성능좋은 하드웨어를 사용할 경우 변복조설비(Modem)없이 디지탈 신호만을 1.5km까지 전송할 수 있다.

동기통신과 비동기통신

직렬통신은 동기(Synchronous)통신과 비동기(Asynchronous)통신으로 구분된다.

(1) 비동기통신

직렬 데이터를 선로를 통해 스타트-스톱 포맷을 가지고 전송한다. 직렬로 전송되는 각 문자(1바이트 데이터)는 '0'상태의 스타트비트로 시작되어 하나 혹은 두개의 스톱비트('1'의 상태)로 끝난다. 송신측과 수신측의 동기는 각 문자단위로 이루어 지므로 전송되는 문자들 사이에 시간간격이 있을 수 있고 이 시간간격동안 선로의 논리상태는 '1'의 상태를 유지해야 한다.

스타트비트와 스톱비트 사이에는 데이터비트와 패리티(Parity)비트가 존재한다. 데이터비트는 7 혹은 8 비트로 구성되는데 아스키코드(ASCII코드)만을 전송할 경우 7비트면 된다. 패리티비트는 홀수(odd), 짝수(even)을 선택하거나 사용하지 않을 수도 있다.

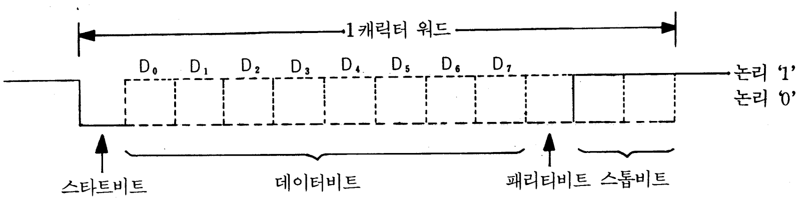
그림 1은 비동기통신방식의 데이터 형식을 나타낸다.

비동기 통신방식에서는 송신하는 데이터 중에 스타트비트 및 스톱비트 등 송신하는 데이터와 전혀 관계가 없는 데이터가 포함된다. 예를 들어 스톱비트를 1비트로, 데이터비트를 8비트로, 패리티 없이 사용할 경우 전체 10비트 중 순수 데이터는 8비트이므로 사용효율은 80%가 된다(비동기방식의 최대 사용효율). 이것은 전용의 동기신호 라인을 사용하지 않고 수신되는 신호 자체로 동기를 잡기 때문이며 사용효율이 나빠지는 것은 불가피하다.

(2) 동기통신

동기통신방식은 비동기통신방식에서 사용효율이 떨어지는 것을 개선하기 위

〔그림 1〕 비동기 통신방식의 데이터 포맷



해 고안된 방식이다. **그림 2**에 동기통신방식의 포맷을 나타냈다.

이 방식에서는 송신측에서 동기클럭신호를 송신하고 수신측에서는 이 클럭신호를 기준하여 데이터를 수신한다.

전송되는 문자들은 스타트비트나 스톱비트 없이 8비트로 구성되며 전송단위가 단일 문자가 아니라 문자들의 집합인 프레임(Frame)이라는 블록이다. 프레임은 연속 비트열이므로 프레임내에서는 문자사이의 시간간격이 허용되지 않는다. 프레임의 개시 시점을 알기 위해 송신하는 데이터속에 그 개시를 나타내는 데이터를 미리 정해 두고(주로 16H) 그 데이터가 나타났을 때를 프레임의 개시로 생각한다.

동기통신방식은 프레임의 구성방식에 따라 분류되는데 주로 이용되는 것은 BSC(Binary Synchronous Communication)방식과 HLDC(High Level Data Link Control)방식이다.

비동기방식은 한 문자마다 동기가 이루어 지므로 동기 하드웨어의 정밀도가 좀 낮아도 쓸 수 있다. 반면 동기방식은 효율은 좋지만 블록단위로 동기가 이루어 지므로 동기 하드웨어가 매우 정교해야 한다.

일반적으로 동기통신방식은 대형기종간의 교신에 사용되고 개인용 컴퓨터는 비동기통신방식을 주로 사용한다. 본고에서는 이후로 비동기방식만을 다루겠다.

보오레이트와 BPS

이 두 용어는 전송속도를 나타내는데 그 정의가 책마다 다른 경우가 있다. 보오레이트(Baud Rate)는 초당 전송되는 비트수를 나타내고(스타트비트, 스톱비트 포함) BPS(Bits Per Second)는 초당 전송되는 데이터비트수 만을 나타낸다(패리티비트 포함).

예를 들어 전송되는 비트의 시간간격이 9.09ms라면 보오레이트는 1/9.09인 110이 된다. 이때 전송형식이 7 데이터비트, 2 스톱비트, 패리티가 존재한다면 BPS는 110×8/11인 80이 된다.

DTE와 DCE

처음 이 용어를 대하는 사람은 그 의미를 잘 이해하기 어렵다. 이 글 전체를 읽고나면 두 용어의 뜻을 파악할 수 있을 것이다.

그림 3에서 멀리 떨어진 두 컴퓨터간의 정보전송은 변복조설비인 모뎀을 거쳐야 한다. 선로가 길어지면 주파수대역폭이 작아지므로 직접 디지탈신호를 전송할 수 없기 때문에 모뎀을 통해 디지탈신호를 아날로그신호로 바꾸어야 한다. 이때 컴퓨터측을 DTE(Data Terminal Equipment)라 하고 모뎀측을 DCE(Data Communication Equipment)라 한다.

그러나 꼭 모뎀만이 아니라 프린터등의 주변장치도 DCE로 사용될 수 있다. DTE와 DCE의 차이는 공통선을 제외한 모든 신호선의 입출력 방향이 서로 반대라는 것이다. 이때 신호선의 명칭은 DTE를 기준으로 하여 정한다. **그림 4**의 경우처럼 트랜스미트 데이터 라인은 DTE쪽에서는 송신신호이나 DCE쪽에서는 수신신호이다. 컴퓨터 중에서도 어떤 기종은 자신의 직렬포트를 DCE형식으로 만든 것도 있다.

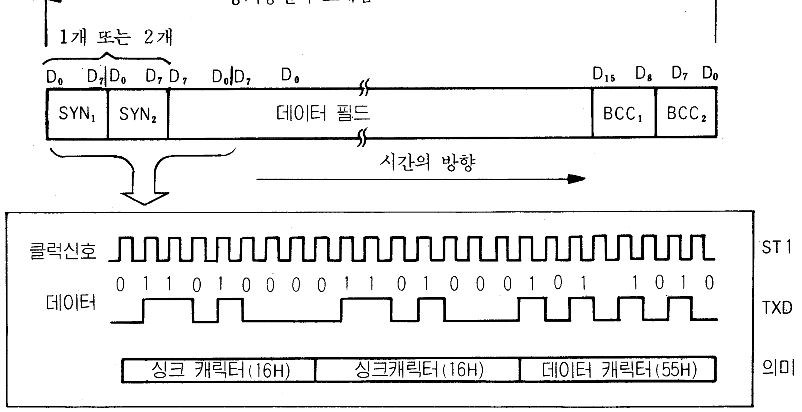
일반적으로 DTE와 DCE는 접속 코넥터의 극성을 달리하고 있다.

DTE와 DCE를 연결할 때는 코넥터 핀을 일대일로 대응시키면 되나 DTE와 DTE끼리의 연결시(예를 들면 모뎀 없이 컴퓨터끼리 직접 연결할 경우) 신호선을 서로 반대로 연결하는 수고를 해야 한다.

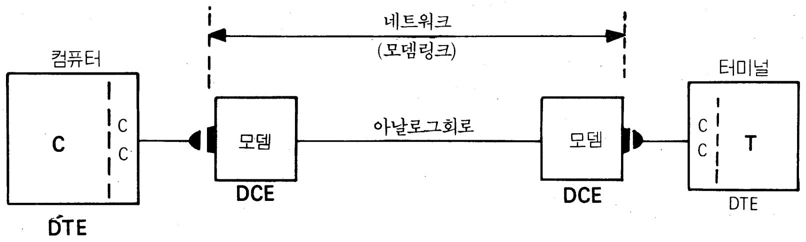
RS-232C 인터페이스

미국의 Electronics Industries Association(EIA)은 한 인터페이스 규격을 만들었는데 이것이 현재까지 가장 널리

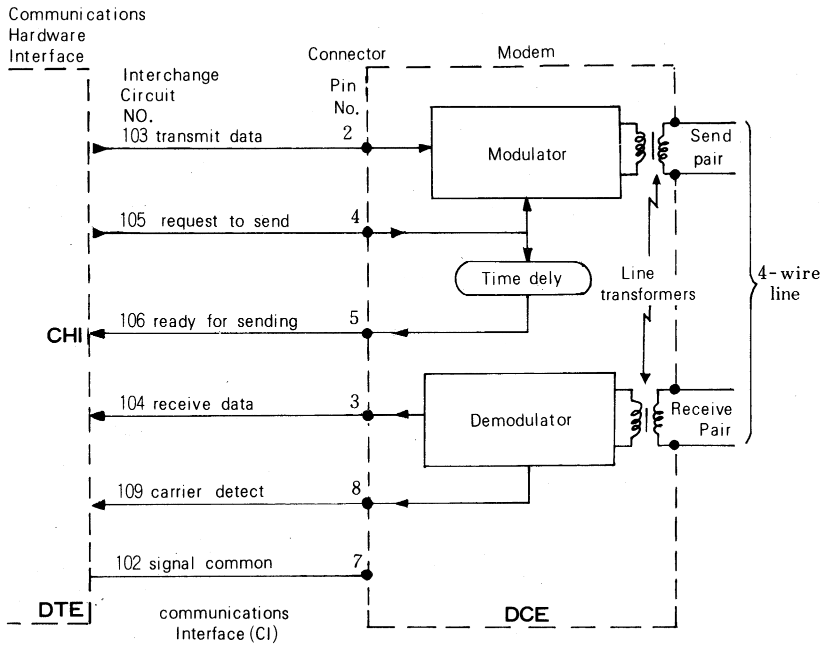
〔그림 2〕 동기통신방식의 데이터 포맷



〔그림 3〕 DTE와 DCE



〔그림 4〕 DTE와 DCE의 신호방향



알려지고 많이 쓰이고 있는 EIA RS-232C 규격이다(RS는 Recommended Standard의 약자이고 C는 개정번호를 나타낸다).

EIA RS-232C 규격은 '최대속도 20,000bit/s로 직렬 이진(Binary) 데이터를 상호교환하는 DTE와 DCE 장비의 불균형접속*(Unbalanced Interface)'을 정의한다.

이 규격으로 동기와 비동기 방식 모두를 사용할 수 있으나 케이블의 길이는 15m(50피트)이내이어야 한다.

이 규격은 3부분으로 세분되는데 각 부분마다 CCITT(International Telegraph and Telephone Consultative Committee)와 ISO(International Organisation for Standardisation)의 규격과 일치한다.

결국 EIA RS-232C 규격은 다음과 같다.

(1) ISO 2110

25핀 DTE/DCE 인터페이스 코넥터의 기계적 특성과 핀배치를 정의(**그림 5** 참고).

통상 DTE는 수(male) 코넥터, DCE는 암(Female) 코넥터를 사용한다.

(2) CCITT V.28

전기적 특성을 정의.

최대전송속도는 20,000bit/s이며 케이블 길이는 최대 15m이다.

〈송신측〉

① 최대출력전압(무부하) 25V(절대값)

② 최소출력전압(부하) 5~15V(절대값, 부하 3~7kΩ에 대해)

③ 전원 OFF시 최소 출력저항 300Ω

④ 단락시의 최대 출력전류 500mA(절대값)

⑤ Slew Rate 최대 30V/μs

〈수신측〉

① 입력임피던스 3~7kΩ

② 입력문턱전압 3V(절대값)

③ 입력전압 최대 25V(절대값)

〈논리〉; 부논리

'1' → −V(통상 −12V)

'0' → +V(통상 +12V)

(3) CCITT V.24

회로기능을 정의

① **회로 AA(Protective Ground)**

보호용 접지로서 장치의 프레임에 접속.

② **회로 AB(Signal Ground/Common Return)**

신호용 접지 또는 공통 리턴선으로 인터페이스 회로에 모두 공통된다.

③ **회로 BA(Transmitted Data)**

송신 데이터로서 DTE에서 발생한 데이터를 DCE에 송신하는 것이다. DTE는 CA, CB, CC, CD 회로가 모두 ON 상태로 되기까지 데이터를 송신해서는 안되도록 되어 있으며 데이터가 보내지지 않을 때는 논리 '1'이 되어야 한다.

④ **회로 BB(Received Data)**

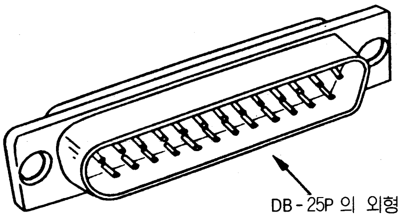
DCE에서 보내져 오는 수신 데이터

***불균형 접속(Unbalanced Interface)과 균형 접속(Balanced Interface)**

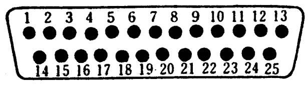
• **불균형 접속**: 모든 신호가 하나의 공통선을 공유; 10개 신호가 있다면 라인은 11개 필요.

• **균형 접속**: 각 신호선은 자신의 복귀선(Return Wire)을 가진다.; 신호가 10개이면 라인은 20개 필요.

〔그림 5〕 25핀 코넥터



DB-25P 의 외형



수 코넥터 DB-25P

를 DTE에 전송하는 회로이다. 이상의 AB, BA, BB 회로가 데이터 전송의 기본 신호회로로서 기본적으로는 이것만 있으면 데이터의 송수신이 가능하지만 실제로 데이터 전송을 하는 경우에는 아래에 설명하는 제어신호에 의해 데이터의 송신절차를 정해야 한다.

⑤ 회로 CA(Request to Send)

모뎀의 송신기능을 제어하는 회로로서 ON 상태에서 모뎀은 송신모드가 되어 캐리어(반송파)를 송출한다.

⑥ 회로 CB(Clear to Send)

모뎀이 데이터를 송신할 수 있는지의 여부를 나타내는 것으로 이 회로가 ON이면 송신가능이다.

⑦ 회로 CC(Data Set Ready)

모뎀(DCE측)과 DTE 사이에서 데이터의 송수신을 할 경우 모뎀이 동작 가능한 상태인가를 나타내는 회로이다. 이 회로가 OFF인 경우에는 모뎀이 시험상태나 전원이 끊긴 상태이다.

⑧ 회로 CD(Data Terminal Ready)

회선과의 접속제어용 회로로서 ON인 경우에는 DTE가 송수신 가능한 상태이다.

⑨ 회로 CE(Ring Indicator)

다이얼업 모뎀의 경우 모뎀이 호출신호(전화선에서)를 수신하고 있는지의 여부를 나타내는 회로이다. ON인 경우에는 호출신호 수신중임을 나타낸다.

⑩ 회로 CF(Carrier Detect)

캐리어 검출회로로서 수신 캐리어가 규격내인가를 검출한다.

⑪ 회로 CG(Signal Quality Detector)

수신 데이터에 오류가 생길 가능성이 있는지의 여부를 나타내는 회로로서 OFF인 경우에는 오류의 가능성이 있다는 것을 DTE측에 알린다.

⑫ 회로 CH(Data Signal Rate Selector…DTE)

전송속도의 선택을 하는 신호이다. 이 경우에는 DTE측에서 DCE에 대해 주어진다. ON인 경우에는 더 빠른 속도를 선택하고 OFF인 경우에는 낮은 속도를 선택한다. 이 회로를 이용하면 전송선로의 품질에 따라 최적의 속도로 전송하는 것이 가능하다.

⑬ 회로 CI(Data Signal Rate Selector…DCE)

회로 CH와 같은 기능이지만 회로 CH와는 반대로 신호의 방향이 DCE에서 DTE 쪽으로 된다.

⑭ 회로 DA(Transmitter Signal Element Timming…DTE)

모뎀의 동기신호로서 동기식의 모뎀에만 사용된다. 이 신호의 폴링에지(Falling Edge)에서 데이터를 검출한다.

⑮ 회로 DB(Transmitter Signal Element Timming…DCE)

DCE에서 DTE로의 동기신호이다. 회로 DA와는 반대로 라이징 에지(Rising Edge)에서 데이터를 검출한다.

⑯ 회로 DD(Receiver Signal Element Timming…DCE)

모뎀의 동기신호로서 DB와 같은 기능이지만 동기신호의 폭이 등간격이다.

⑰ 회로 SBA(Secondary Transmitted Data)

제 2의 송신 데이터 채널로서 회로 BA와 같은 기능을 나타낸다.

⑱ 회로 SBB(Secondary Received Data)

제 2의 수신 데이터 채널로서 회로 BB와 같은 기능을 나타낸다.

⑲ 회로 SCA(Secondary Request to Send)

제 2채널의 송신요구 회로로서 회로 CA와 같은 기능을 나타낸다.

⑳ 회로 SCB(Secondary Clear to Send)

제 2채널 CB

㉑ 회로 SCF(Secondary Received Line Signal Detector)

제 2채널 CF.

EIA RS-232C 규격과 CCITT V.24 권고안의 주된 차이는 EIA는 각 교환회로의 명칭을 영문자로 나타내는데 반해 CCITT는 숫자 기호를 쓰는 것이다. 예를 들면 송신 데이터 회로(Transmit Data Interchange Circuit)를 RS-232C 규격은 'BA'라고 나타내는 반면 CCITT에서는 '103'으로 나타낸다.

표 1은 RS-232C 규격 접속 코넥터의 핀 배치와 할당된 신호들을 나타낸다.

RS-232C 규격에서 각 회로의 명칭은 회로의미를 나타낸 말의 약자가 아니라 고유코드이므로 명칭만 보고는 무슨 뜻인지 알 수 없다. 따라서 본고에서는 이후로 모든 명칭을 알기 쉽게 약자로 풀이하겠다. 즉 코넥터 2번 핀의 명칭을 'BA'가 아닌 'TXD'(Transmitted Data의 약자)로 호칭하는 식이다.

RS-232C 인터페이스 규격은 원래 컴퓨터간의 직접 전송을 위해 마련된 것이 아니고 컴퓨터와 모뎀간 즉 DTE와 DCE간의 전송에 관한 규격이다.

모뎀은 컴퓨터처럼 언제라도 데이터를 송수신할 수 있는 것이 아니고 외부선로(공중전화망 등)의 환경에 따라 복잡하게 상태가 변화하며 이를 DTE측에 알려 주어야 한다. 그렇기 때문에 20여개의 신호회로가 필요하게 된다. 그러나 모뎀을 사용할 지라도 모든 신호회로를 다 쓰는 일은 고급 동기모뎀의 경우외에는 별로 없고 대부분의 경우 RS-232C 규격을 간략화하여 다음과 같이 일부만을 사용한다(번호는 코넥터 핀번호, 괄호안은 정식규격명칭).

〔표 1〕 V. 24/V. 28(RS-232C) 인터페이스

25-pin Connector Pin No.	CCITT Circuit Number	EIA RS-232 C Ref	Description of Interchange Circuit	Direction to DCE	Direction to DTE
1	–	AA	Protectve Ground		
2	103	BA	Transmitted Data(TD)	X	
3	104	BB	Received Data(RD)		X
4	105	CA	Request To Send(RTS)	X	
5	106	CB	Ready For Sending[Clear To Send] (CTS)		X
6	107	CC	Data Set Ready(DSR)		X
7	102	AB	Signal Ground or Common Return		
8	109	CF	Data Channel		X
			Received Line Signal Detector(RLSD)		
9	–	–	N (N=reserved for)		
10	–	–	N (National Use)		
11	126	–	Select Transmit Frequency	X	
12	122	SCF	Backward Channel(Secondary) RLSD		X
13	121	SCB	Backward Channel Ready(Secondary CTS)		X
14	118	SBA	Transmitted Backward Channel Data	X	
			(Secondary Transmitted Data)		
15	114	DB	Transmitter Signal Element Timing		X
16	119	SBB	Received Backward Channel Data		X
			(Secondary Received Data)		
17	115	DD	Receiver Signal Element Timing		
18	141	–	Local Loopback(Loop 3)	X	
19	120	SCA	Transmit Backward Channel Line Signal	X	
			(Secondary Request To Send)		
20	108/1	–	Connect Data Set To Line(CDSTL) *	X	
	108/2	CD	Data Terminal Ready(DTR) *	X	
21	140	–	Remote Loopback(Loop 2) *	X	
	110	CG	Data Signal Quality Detector *		X
22	125	CE	Calling(Ring) Indicator(RI)		X
23	111	CH	Data Signalling Rate Selector *	X	
	112	CI	Data Signalling Rate Selector *		X
24	113	DA	Transmitter Signal Element Timing	X	
25	142	–	Test Indicator		X
Pin Assignment Variations often used by Modem Manufacturers					
9	–	–	Positive Test Voltage(USA Modems)		
10	–	–	Negative Test Voltage(USA Modems)		
9	116	–	Select Standby	X	
11	116	–	Select Standby	X	
24	116	–	Select Standby	X	
10	117	–	Standby Indiccator		X
25	117	–	Standby Indicator		X
11	110	CG	Data Signal Quality Detector		X
12	110	CG	Data Signal Ouality Detector		X
24	110	CG	Data Signal Quality Detector		X
25	110	CG	Data Signal Quality Detector		X

* Selectable

① **FG** : Frame Ground (AA)
보호용 접지. 없어도 무방.

② **TXD** : Transmitted Data (BA)
DTE → DCE로의 전송라인.

③ **RXD** : Received Data (BB)
DCE → DTE로의 전송라인.

④ **RTS** : Request to Send (CA)

⑤ **CTS** : Clear to Send (CB)

DTE가 DCE로 전송하려면 RTS선을 '0'으로 한다(부논리). DCE는 DTE가 송신을 원하는 것을 알고 외부 선로에 반송파를 내보낸다. DCE가 DTE로부터 수신할 수 있는 상태가 되면 DCE는 CTS선을 '0'로 하여 DTE에 송신을 허가한다. DTE는 DCE로 TXD 라인을 통해 전송하고 DCE는 DTE에서 받은 데이터를 변조하여 외부선로에 내보낸다. 송신이 끝난 DTE는 RTS선을 '1'로 하여 수신상태로 된다(**그림 6** 참고).

⑥ **DSR** : Data Set Ready (CC)
'0' 상태이면 DCE전원 ON 상태임을 DTE에 알린다.

⑦ **SG** : Signal Ground (AB)
공통선. 반드시 필요.

⑧ **CD** : Carrier Detect (CF)
'0' 상태이면 외부선로에서 정상적인 반송파가 검출되고 있음을 DTE에 알린다.

⑨ **DTR** : Data Terminal Ready (CD)
'0'상태이면 DTE전원이 ON 상태임을 DCE에 알린다.

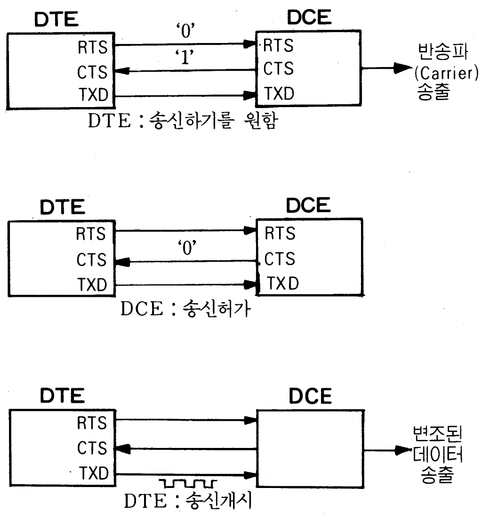
⑩ **RI** : Ring Indicator (CE)
다이얼-업 모뎀을 사용할 경우에만 필요. '0' 상태이면 외부선로인 전화선으로 호출되고 있음을 DTE에 알린다.

사실 IBM-PC를 포함한 대부분의 개인용 컴퓨터에 설치된 RS-232C 인터페이스는 위의 10개 신호회로나 혹 그 이하 만을 사용하고 있다.

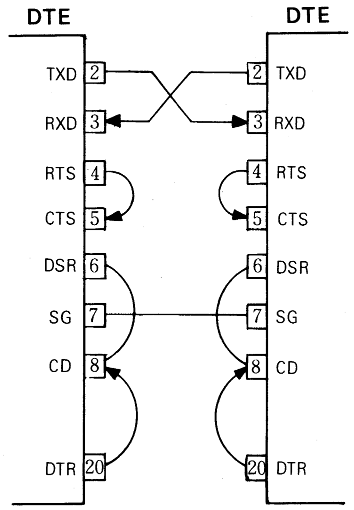
RS-232C 인터페이스를 자유자재로 이용하려면 최소한 위 10개 신호회로의 의미는 잘 알아야 한다. 처음 접하는 사람은 특히 RTS, CTS 신호회로의 의미를 이해하기 힘들어 하는데 이 부분은 그림 6에 잘 나타나 있다.

두 컴퓨터간을 모뎀없이 직접 접속할 경우, 즉 DTE와 DTE의 접속인 경우 **그림 7**과 같이 연결해야 한다. 이렇게 하는 이유는 조금만 생각하면 곧 이해할 수 있을 것이다. 이런 방식으로 DTE-DTE간을 연결하는 케이블을 'Null-

〔그래 6〕 RTS, CTS 핸드세이킹



〔그림 7〕 Null-Modem Cable을 이용한 DTE-DTE간 접속



Modem 케이블'이라 한다.

RI입력은 아무 것도 연결하지 않고 놀리면 된다.

통신형태

반 2 중(Half Duplex)과 전 2 중(Full Duplex)

〈정의 1〉

반2중 : 한번에 한 방향으로만 전송 가능하다. 송신을 하고 있는 동안에는 수신을 할 수 없고 수신을 하고 있는 동안에는 송신을 할 수 없는 방식. 이런 형태의 모뎀을 '2선 모뎀'이라 한다. 많이 쓰이지 않는 방식이다.

전2중 : 동시에 송수신이 가능하다.. '4선 모뎀'이 전 2 중 형태의 모뎀이다.

정의 1이 원칙적인 정의이나 최근의 개인용 컴퓨터는 인터럽트를 유효하게 이용하여 모두 전 2 중으로 하는 것이 가능하여 이와 같은 정의로는 의미가 없으므로 새로운 정의 2가 오히려 적절하다.

〈정의 2〉

반2중 : 터미널의 키보드에서 입력된 문자는 메인으로 전송되며 동시에 터미널의 화면에 출력된다. 키보드의 데이터가 반드시 메인에 전송되었는지의 여부는 불명이다.

전2중 : 터미널의 키보드에서 입력된 문자는 터미널 화면에 표시되지 않고 곧장 메인으로 전송되며 메인에서는 이 데이터를 다시 터미널로 보내고 터미널 화면에 표시된다(에코우 백 기능).

통신로의 이상이 확인되지만 통신에 소요되는 시간이 반2중에 비해 많이 걸린다. 전 2 중 메인 컴퓨터에 반 2 중 터미널을 부착하면 키보드입력시 화면에 글자가 2 개씩 찍히게 된다. 반 2 중 메인에 전 2 중 터미널을 연결하면 키보드로 입력한 데이터가 화면에 나타나지 않는다.

CP/M 등의 OS에서는 터미널과 본체간의 데이터전송은 전2중 방식을 취하고 있지만 대형 컴퓨터의 공중전화망에 의한 터미널은 일반적으로 반2중 방식을 취하고 있는 경우가 많다.

IBM-PC와 RS-232C 인터페이스

IBM-PC/XT/AT 기종은 RS-232C 인터페이스가 내장되어 있거나, 내장되지 않았더라도 손쉽게 인터페이스 카드를 구할 수 있다.

MS-DOS상에서는 com1 : ～com4 : 까지 4 개의 직렬포트를 지정할 수 있으며 Device File AUX는 특별한 지정이 없다면 com1에 할당된다. DOS 내장명령어 MODE로 직렬포트의 초기화를 할 수 있으며 그 형식은 다음과 같다.

```
MODE <port> :<baud>,<parity>,<data>,<stop>
```

예를 들어 com2 포트를 보오레이트 1200, 짝수패리티, 7 데이터비트, 2 스톱비트로 초기화 하려면

```
MODE com2 : 1200,e,7,2
```

를 DOS 레벨에서 입력하면 된다. 데이터 전송은 간단히 Copy 명령을 사용할 수도 있고 'Cross Talk' 등의 통신프로그램을 이용할 수도 있다.

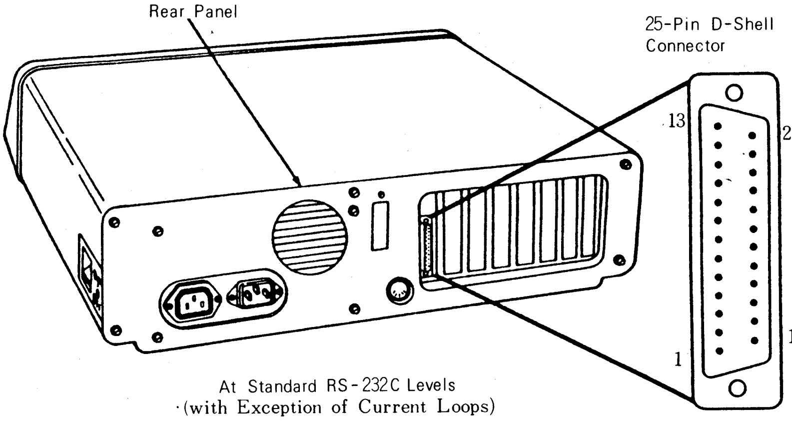
IBM-PC의 RS-232C 인터페이스를 위한 하드웨어는 **그림 8**과 같다.

애플II의 RS-232C 인터페이스

애플II에는 RS-232C 인터페이스가 내장되어 있지 않기 때문에 인터페이스 카드를 추가해야 한다.

애플II용 RS-232C 인터페이스 카드로 가장 유명한 것은 '캘리포니아 컴퓨터 시스템'사의 '모델 7710'이다. 이것

〔그림 8〕 IBM-PC RS-232-C인터페이스



External — Asynchronous Communications Adapter (RS-232C)

Description	Pin
NC	1
Transmitted Data	2
Received Data	3
Request to Send	4
Clear to Send	5
Data Set Ready	6
Signal Ground	7
Received Line Signal Detector	8
+Transmit Current Loop Data	9
NC	10
-Transmit Current Loop Data	11
NC	12
NC	13
NC	14
NC	15
NC	16
NC	17
+Receiv Current Loop Data	18
NC	19
Data Terminal Ready	20
NC	21
Ring Indicator	22
NC	23
NC	24
—Receive Current Loop Return	25

은 애플 CP/M에서 디스크콘트롤러 카드나 바이덱스 카드처럼 표준 주변장치로 인정하고 있다.

그러나 불행히도 이것은 국내에서 입수가 불가능 하므로 동일한 기능을 수행하는 카드를 손수 제작해야 한다.

그림 9가 제작할 카드의 회로도이며 이 회로도대로 제작한 것이 **사진 1**의 인터페이스 카드이다. 사진에서 보는 바와 같이 회로 자체는 매우 간단하다. 이 회로의 중심은 6850 LSI 칩이다. 6850은 ACIA(Asynchronous Serial Interface Adapter)라고도 하는 비동기 통신 전용 칩이다. 원래는 68계열 CPU의 주변 칩인데 6502도 그대로 이용할 수 있다. TTL 레벨과 RS-232C 규격 레벨의 변환은 1488과 1489로 한다. 1488은 RS-232C 라인드라이버전용 칩이고 1489는 RS-232C 라인리시버전용 칩이다. 각각 4개의 게이트를 가지고 있으므로 1488과 1489 각각 1개씩이면 족하다. 1488의 +V, −V에는 각각 +12V, −12V를 사용한다.

74HC4060은 클럭발생기로 쓰인다. 이 칩의 여러 출력중 하나를 딥스위치로 선택하여 6850의 TXC(Transmit Clock)와 RXC(Receive Clock)에 공급한다. 6850은 이 신호를 16분주 혹은 64분주하여 보오레이트 클럭으로 사용한다. 16분주시 보오레이트 300으로 송수신하려면 TXC, RXC에 공급되는 클럭의 주파수는 300×16=4800이다. 74HC4060의 출력중 주파수 4800Hz인 것은 클럭 주파수 4915.2kHz를 10분주한 출력인 Q10이다. 딥스위치로 Q10 출력을 선택하면 전송속도는 300 보오레이트가 된다. 같은 원리로 Q9로는 600, Q8로는 1200, Q7로는 2400 보오레이트를 선택할 수 있다. 딥스위치는 반드시 4개중 하나만 ON 되어야 한다.

크리스탈 주변의 RC회로는 안정된 발진을 위한 것이다. 크리스탈 케이스에 납땜할 수 있는 부분이 있으므로 크리스탈 케이스는 접지시킨다.

6850의 $\overline{IRQ}$ 출력을 직접 6502 $\overline{IRQ}$에 연결하지 않고 회로도와 같이 한 것은 Dasy-Chain 체계를 유지하기 위한 것이다. 6850 IRQ 출력은 오픈 드레인 형태이므로 꼭 풀업시킨다.

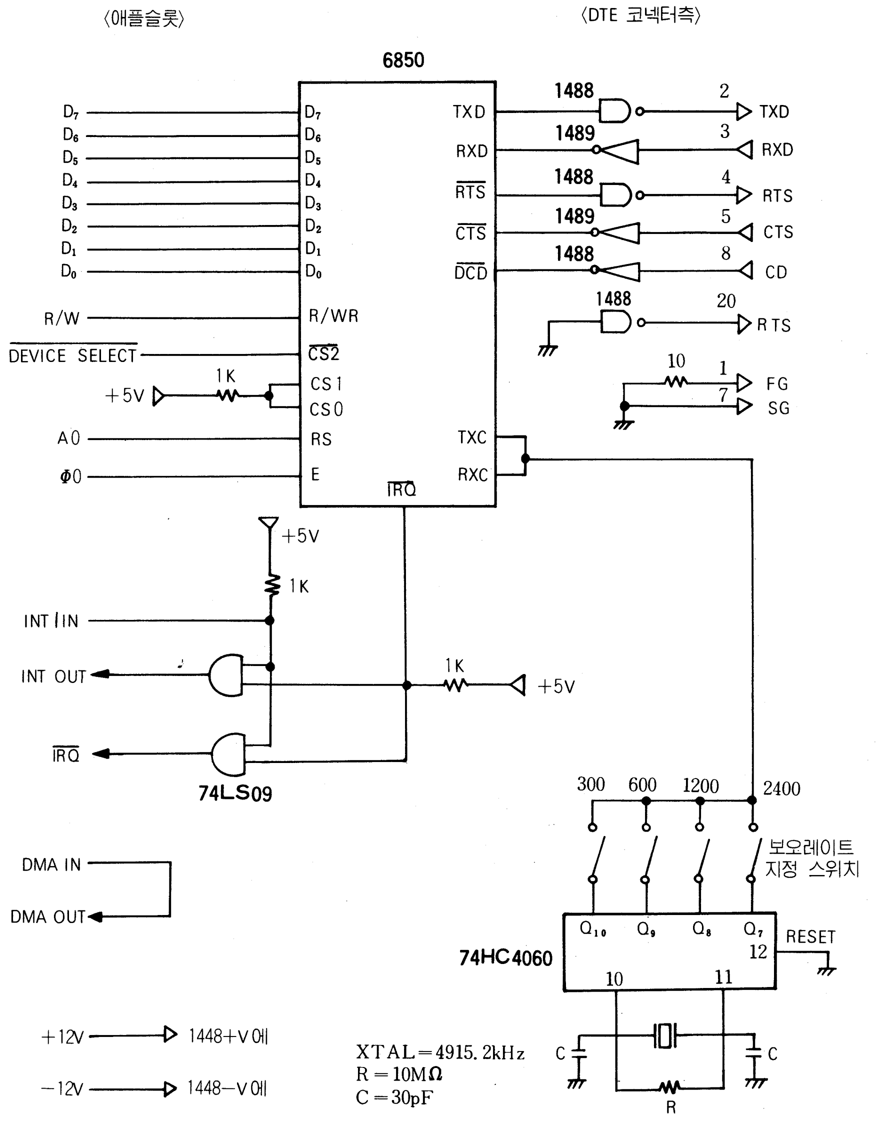
하드웨어에 약간의 지식이 있는 사람은 쉽게 완성시킬 수 있을 것이다.

이 카드를 제대로 이용하려면 ACIA 6850에 관해 알아야 한다. 6850에는 4개의 8비트 레지스터들이 있는데 그 어드레스 할당은 다음과 같다.

$C0X0(WR)=command
$C0X0(RD)=status
$C0X1(WR)=transmit
$C0X1(RD)=receive

X는 〈슬롯번호+8〉을 나타낸다. Co-

〔그림 9〕 APPLEⅡ RS-232C 인터페이스



mmand 레지스터와 Transmit 레지스터는 CPU가 라이트만 가능하고 Status 레지스터와 Receive 레지스터는 CPU가 리드만 가능하다.

표 2와 **표 3**은 6850 Command 레지스터와 Status 레지스터의 기능을 요약한 것이다.

6850에는 하드웨어 리세트기능이 없고 Command 레지스터의 Bit0, Bit1을 '//'로 하여 리세트 동작을 수행한다. 리세트시 모든 레지스터 값은 초기화되나 Command 레지스터의 나머지 비트들은 영향을 받지 않는다. 6850을 사용할 때는 가장 먼저 리세트 동작을 수행해야 한다.

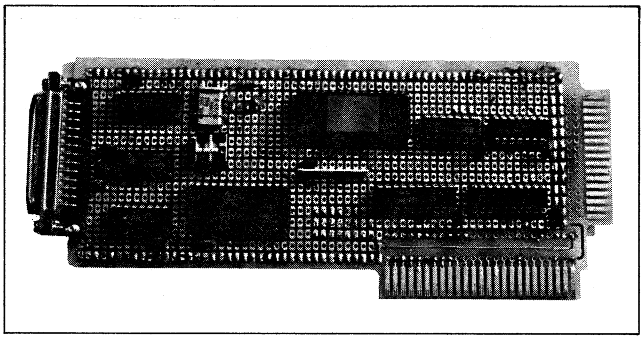
그림 7과 같은 Null-Modem 케이블을 사용할 경우 RTS를 항상 '0'으로 해두면 항상 송·수신 가능한 상태가 된다.

6850을 제어하는 프로그램을 작성할 때 인터럽트를 이용하면 효율적이지만 꼭 인터럽트를 이용하지 않아도 된다.

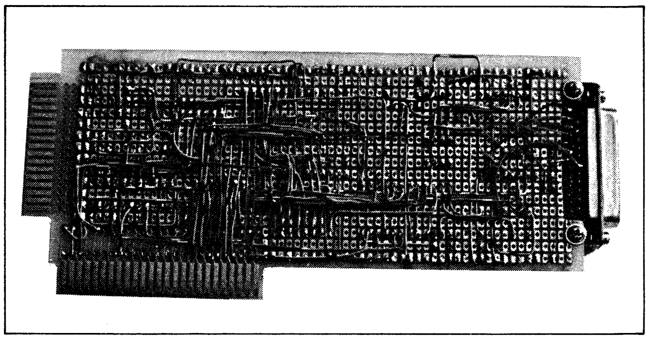
필자는 인터럽트를 이용하지 않고 Status 레지스터의 비트를 검사하는 방법으로 프로그램을 작성했다. 6850을 통해 데이터를 송·수신하는 알고리즘은 **그림 10**과 같다.

데이터 전송을 수행하는 작업은 CP/M OS상에서 행한다. 6502를 사용하는 표준 DOS 3.3보다는 CP/M이 모든 면에서 월등히 우수한 환경이기 때문이다. 이 인터페이스 카드는 6850 레지스터

〔사진 1〕 완성된 애플용 RS-232C 인터페이스카드의 실물사진



(a) 앞면



(b) 뒷면

〔표 2〕 6850 Command레지스터

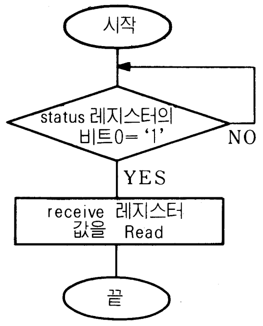
ACIA COMMAND		
7 6 5 4	3 2 1 0	
××××	××0 0	The clock is 1× the baud rate
××××	××0 1	The clock is 16× the baud rate
××××	××1 0	The clock is 64× the bad rate
××××	××1 1	ACIA Master Reset
×××0	0 0××	7data+Even Parity+2stop bits
×××0	0 1××	7data+Odd parity+2stop bits
×××0	1 0××	7data+Even parity+1stop bits
×××0	1 1××	7data+Odd parity+1 stop bit
×××1	0 0××	8data+No parity+2stop bits
×××1	0 1××	8data+No parity+1stop bit
×××1	1 0××	8data+Even parity+1stop bit
×××1	1 1××	8data+Odd parity+1stop bit
×0 0×	××××	$\overline{\text{RTS}}$= '0'. Disable transmit interrupt
×0 1×	××××	$\overline{\text{RTS}}$= '0'. Enable transmit interrupt
×1 0×	××××	$\overline{\text{RTS}}$= '1'. Disable transmit interrupt
×1 1×	××××	$\overline{\text{RTS}}$= '0'. Disable transmit interrupt Transmit break signal
0×××	××××	Disable Receive Interrupt
1×××	××××	Enable Receive Interrupt When : Receiver data register full Receiver data overrun CD signal inactive

〔표 3〕 6850 Status 레지스터

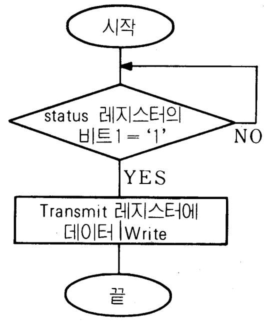
7	6	5	4	3	2	1	0
IRQ	PE	OVRN	FE	$\overline{\text{CTS}}$	$\overline{\text{DCD}}$	TDRE	RDRF

	〔'1' 의 상태일 때〕
BIT 0	Receive data is ready for the computer
BIT 1	The transmit register can accept data
BIT 2	RS-232C CD inactive ; Bit0 → '0'
BIT 3	RS-232-C CTS inactive; Bit 1→ '0'
BIT 4	Received data improperly framed
BIT 5	Data received before previous byte read
BIT 6	Parity Error in received data
BIT 7	ACIA-generated trasmit or received interrupt

〔그림 10〕 송·수신 알고리즘



(a) 수신알고리즘



(b) 송신알고리즘

들의 어드레스 할당이 CP/M이 표준 입출력장치로 정한 CCS7720의 그것과 동일하기 때문에 사용자가 일일이 프로그램을 작성하지 않고도 BDOS CALL을 이용해 직렬포트의 입출력을 수행할 수 있다. BDOS CALL 4번은 RS-232C 출력이고, BDOS CALL 3번은 RS-232C 입력이다.

CP/M에서 Logical Device PUN:은 RS-232C 출력으로, Logical Device RDR:은 RS-232C 입력으로 정의되며, PUN:에는 Physical Device PTP:, UP1:, UP2:중 하나를 할당할 수 있고 RDR:에는 Physical Device PTR:, UR1:, UR2:중 하나를 할당할 수 있다. 즉 CP/M에서는 3개의 RS-232C 포트를 이용할 수 있는 셈이다.

CP/M OS상에서 STAT VAL: 명령을 수행시키면 Logical Device CON:, RDR:, PUN:, LST: 각각에 대해 할당할 수 있는 Physical Device의 목록이 나열된다. STAT DEV: 명령을 수행시키면 현재 Logical Device에 할당되어 있는 Physical Device의 목록이 나열된다.

애플 CP/M의 경우 주의해야 할 것이 있다. 독자는 CP/M OS체계를 잘 알고 있다고 가정한다. STAT DEV: 명령으로 초기상태 즉 사용자가 특별히 변경하지 않은 상태의 Physical Device 할당내용을 보면 다음과 같다.

CON : — CRT :
RDR : — PTR :
PUN : — PTP :
LST : — LPT :

그런데 어드레스 03H에 위치한 IOBYTE의 내용을 조사해 보면 RDR:이

PTP:에 할당된 것이 아니라 콘솔 디바이스 TTY:에 할당된 것으로 나타난다. 애플 CP/M의 시스템 버그가 아닌가 싶다. 그렇다면 Physical Device PTR:은 TTY:와 동일하므로 이 상태로는 BDOS를 이용한 전송이 불가능하다. 이때는

```
STAT RDR : = UR1 :
```

명령으로 디바이스를 합당하게 재할당해주어야 한다.

UR1:과 UR2:는 다르지만 UP1:과 UP2:는 동일한 디바이스이므로 실지 유효한 디바이스 할당은 다음과 같다.

```
RDR : - UR1 : , UR2 :
PUN : - PTP : , UP1 : (or UP2:)
```

어드레스 0F380H~0F395H의 영역은 I/O 벡터 테이블이라 하는데 각 물리 디바이스들의 엔트리 포인트를 담고 있다(**표 4** 참고).

0FE6FH의 RS-232C 입력루틴과 0FE69H의 RS-232C 출력루틴을 조사해 보면 코드가 지나치게 길다. 이 정도면 차라리 직접 입출력루틴을 작성하는 것이 나을 것 같아 그림 10의 알고리즘을 토대로 UR2:와 UP2:를 위한 프로그램을 작성했다.

애플 CP/M은 어드레스 0F200H~0F37FH의 영역을 사용자의 I/O 디바이스 제어루틴을 로드할 수 있도록 해놓았다(**표 5** 참고).

리스트 1과 **리스트 2**가 각각 UR2:, UP2: 디바이스를 위한 루틴이다. 이들은 MACRO-80 어셈블러의 소스형식으로 작성되었고 어셈블과 링크 작업을 거쳐 .COM 형식의 실행가능한 화일로 만든후 CP/M 시스템디스켓에 있는 유틸리티 프로그램인 CONFIGIO. BAS을 이용해 디스크의 시스템 영역에 세이브한다. 이 프로그램들의 선두에 이상한 코드들이 나열되어 있는 이유는 CONFIGIO 프로그램이 요구하는 포맷에 맞추기 위한 것이다.

CONFIGIO를 실행시킨후 메뉴의 3번 'LOAD USER I/O DRIVER SOFTWARE'기능을 선택하면 화일명을 물어온다. 우선 'UR2.COM'을 입력하면 UR2.COM이 메모리 0F280H 이후에 로드되고(표 5 참고) I/O 벡터 테이블의 7번 'Reader Input#2'의 내용이 0F280H로 바뀐다. 메뉴로 돌아와 다시 3번을 선택하고 'UP2.COM'을 입력하면 마찬가지로 UP2.COM이 앞서 로드한 UR2.COM에 연속되는 지점인 0F28AH 이후에 로드되고 I/O 벡터 테이블의 9번 'PUNCH Output#2'의 내용이 0F28AH로 변경된다.

메뉴의 4번 'Read/Write I/O Configuration Block'기능을 선택하면 변경된 시스템 프로그램을 디스켓에 저장할 수 있다.

이후로는 사용자가 물리 디바이스 UR2:나 UP2:를 사용할 때마다 이 루틴을 이용하게 된다.

이 새로운 UR1, UR2 루틴은 RS-232C 인터페이스 카드가 슬롯 2번에 설치된 것으로 인식한다. CCS 7720카드도 슬롯 2번에 설치하는 것을 원칙으로 하고 있다.

[표 4] 애플 CP/M I/O벡터테이블

VECTOR번호	어드레스	VECTOR명	디바이스	담긴 어드레스
1	0F380H	Console status		0FD99H
2	0F382H	Console input # 1	TTY : , CRT :	0FB33H
3	0F384H	Console input # 2	UC1 :	0FB38H
4	0F386H	Console output # 1	TTY : , CRT :	0FBB5H
5	0F388H	Console output # 2	UC1 :	0FCB5H
6	0F38AH	Reader input # 1	UR1 :	0FE6FH
7	0F38CH	Reader input # 2	UR2 :	0FE6FH
8	0F38EH	Punch output # 1	PTP :	0FE69H
9	0F390H	Punch output # 2	up1 : , up 2 :	0FE69H
10	0F392H	List output # 1	LPT :	0FE55H
11	0F394H	List output # 2	UL1 :	0FE55H

[표 5] IOCB 유저 패치영역

어드레스	할당된 슬롯	할당된 논리 디바이스
0F200H-0F27FH	1	LST :
0F280H-0F300H	2	PUN : & RDR :
0F300H-0F37FH	3	CON :

CONFIGIO.BAS 프로그램을 사용해서 새루틴 UR2.COM과 UP2.COM을 시스템 프로그램 영역에 삽입하는 작업과 같은 것을 패치(Patch)라 한다.

패치 작업이 끝났으면 CP/M OS상에서 STAT 프로그램을 사용하여 다음과 같이 논리 디바이스 RDR:과 PUN:에 새 물리 디바이스 UR2:와 UP2:를 할당한다.

```
STAT RDP : = UR2 :
STAT PUN : = UP2 :
```

이제 부터는 BDOS 시스템 콜(3번과 4번)을 이용한 RS-232C 입출력은 어드레스 0F280H의 UR2.COM과 어드레스 0F28AH의 UP2.COM을 통해 이루어 진다. 이때 6850의 초기화 부분에서 6850 Command 레지스터에 16진수 11이 넣어지므로 직렬전송 데이터의 포맷은 다음과 같다.

```
8 data , NO parity , 2 stop bits
```

데이터 비트를 8비트로 함으로서 ASCII 코드만이 아닌 모든 1바이트 데이터를 전송할 수 있다(아스키코드는 7

〔리스트 1〕 새로운 RS-232-C 입력루틴 UR2. MAC

```
;--------------------------------
;  RS-232-C USER IO DRIVER 1.0
;  PHYSICAL DEVICE UR2:
;         1988.10.24
;--------------------------------
ORIGIN  EQU     0F280H
CKINIT  EQU     0F294H          ;ROUTINE IN UP2.MAC
STATUS  EQU     0E0A0H
CONTRL  EQU     0E0A0H
DATA    EQU     0E0A1H
        .Z80
        ASEG
        ORG     0100H
        DB      1               ;MAKE 1 PATCH
        DW      ORIGIN          ;REAL START ADDRESS
        DW      LAST-RDR        ;CODE LENGTH
        DB      1               ;PATCH TYPE 1
        DB      7               ;PATCH READER INPUT #2
        DW      RDR             ;NEW CONTENTS OF VECTOR

        .PHASE  ORIGIN
RDR:    CALL    CKINIT          ;ALREADY INITIALIZD?
CHKRDR: LD      A,(HL)          ;HL=STATUS
        RRA                     ;[C]<-M0
        JR      NC,CHKRDR       ;REPEAT UNTIL INPUT
        INC     L               ;HL=DATA
        LD      A,(HL)
        RET

LAST:   .DEPHASE
        END
```

〔리스트 2〕 새로운 RS-232-C 출력루틴 UP2. MAC

```
;--------------------------------
;  RS-232-C USER IO DRIVER 1.0
;  PHYSICAL DEVICE UP2:
;         1988.10.24
;--------------------------------
ORIGIN  EQU     0F28AH
STATUS  EQU     0E0A0H
CONTRL  EQU     0E0A0H
DATA    EQU     0E0A1H
        .Z80
        ASEG
        ORG     0100H
        DB      1               ;MAKE 1 PATCH
        DW      ORIGIN          ;REAL START ADDRESS
        DW      LAST-PUN        ;CODE LENGTH
        DB      1               ;PATCH TYPE 1
        DB      9               ;PATCH PUNCH OUTPUT #2
        DW      PUN             ;NEW CONTENTS OF VECTOR

        .PHASE  ORIGIN

PUN:    CALL    CKINIT
CHKTDR: BIT     1,(HL)          ;M1=TDRE
        JR      Z,CHKTDR
        INC     L
        LD      (HL),C
        RET

CKINIT: LD      HL,STATUS
        LD      A,(VEC)
        CP      65H             ;ACIA INITIALIZED?
        RET     Z
INIT:   LD      A,01010011B     ;RTS='1' &
        LD      (CONTRL),A      ;ACIA MASTER RESET

SETVEC: LD      A,65H
        LD      (VEC),A

        LD      A,00010001B     ;RTS='0'
        LD      (CONTRL),A      ;8 DATA, 2 STOP, NO PARITY
        RET

VEC:    DB      0

LAST:   .DEPHASE
        END
```

비트로 구성되었으므로 데이터 비트를 7비트로 설정하면 최상위 비트가 0인 문자들 만을 전송할 수 있다).

동일 실내에서 컴퓨터간의 직접전송시 에러가 날 확률은 매우 희박하므로 패리티는 사용하지 않는다. 패리티를 사용할 경우 패리티에러 처리루틴을 작성해야 한다. 스톱비트를 2비트로 하면 1비트로 할때 보다 문자전송속도(BPS)는 느려지지만 더 안전하다.

Command 레지스터에 지정한 6850의 동작형태는 다음과 같다.

보오레이트 분주비 : ÷16

RTS : 항상 '0'—Null 모뎀 케이블 사용시 항상 송수신 가능.

인터럽트 : 금지

여러분이 이 카드를 이용한 터미널 시뮬레이션 프로그램을 작성한다면 인터럽트 처리를 이용하면 좋을 것이다. 그러나 필자는 화일전송기능만을 필요로 했기 때문에 번거로운 인터럽트 이용은 하지 않았다.

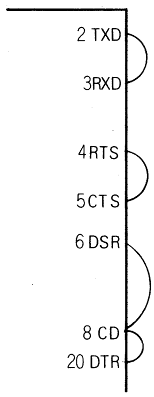
카드가 완성되었으면 테스트를 할 차례다. 가장 많이 쓰이는 루프백(Loop Back) 테스트 방법을 사용한다. 루프백 테스트란 컴퓨터가 자신의 송신채널로 송신한 데이터를 자신의 수신채널로 되받아 확인해 보는 것이다. 이를 위해 **그림 11**과 같은 수코넥터를 준비한다. 그림과 같이 2번 TXD가 3번 RXD에 연결되어 있으므로 6850의 트랜스미트(Transmit) 레지스터에 넣어 송신한 데이터는 리시브(Receive) 레지스터로 들어오게 된다(인터페이스 카드가 정상적으로 동작한다면).

테스트 프로그램은 **리스트 3**의 파스칼 프로그램이다. 컴파일러는 터보파스칼을 사용한다. 터보파스칼은 CP/M BDOS를 호출해 RS-232C 입출력을 할 수 있는 기능을 갖추고 있다. 입출력문 리드나 라이트에서 AUX 디바이스를 지정하면 데이터 입출력이 콘솔을 통해 이루어지는 것이 아니라 RS-232C 직렬포트를 통해 이루어진다.

예)
```
READ ( AUX ,DATA ) ;
WRITE ( AUX , DATA )
```

물론 파스칼 프로그램이 자체적으로 인터페이스 카드를 제어하는 것이 아니라 BDOS를 호출하고 BDOS는 시스템 프로그램 영역의 UR2.COM, UP2.COM을 호출한다. 이 루프백 테스트 프로그램을 컴파일하고 실행하려면 먼저 완성된 인터페이스 카드가 슬롯 2번에 설치되어 있고 카드의 25핀 수코넥터에 그림 11의 루프백 코넥터가 물려 있어야 한다. 또한 UR2.COM과 UP2.COM이 시스템에 패치되고 논리 디바이스 RDR:에는 물리 디바이스 UR2:가, PUN:에는 UP2:가 할당되어 있어야

〔그림 11〕 루트벡 테스트를 위한 코넥터배선



한다. 디바이스 할당은 STAT 프로그램을 이용하거나 어셈블리어 레벨에서 BDOS 콜 8번 기능을 이용한다.

준비가 끝났으면 루프백 테스트 프로그램을 실행시킨다. 프로그램의 기능은 문자 '0'에서 '9'까지 10개의 문자를 차례로 송신한 직후 수신하여, 송신한 문자와 수신한 문자를 나란히 화면에 출력하는 것이다. '0'에서 '9'까지의 문자가 2개씩 나란히 화면에 출력되면 성공이다.

상대방과 통신을 하다보면 전송속도와 데이터 형식을 변경할 필요가 생긴다. 전송속도는 S1에서 S4까지 4개의 딥스위치로 결정한다. 각 딥스위치가 ON 상태일 때의 전송속도는 다음과 같다.

```
S1 - 300 baud
S2 - 600 baud
S3 -1200 baud
S4 -2400 baud
```

하드웨어를 약간만 변경하면 더 높거나 낮은 속도도 얻을 수 있다.

데이터 형식을 초기치인〈8 데이터비트, 2 스톱비트, NO 패리티〉 이외의 것을 사용하려면 6850 Command 레지스터의 비트 4, 3, 2의 내용을 알맞게 수정해야 한다. MS-DOS는 DOS 레벨에서 내장명령어 MODE로 RS-232C 포트를 알맞은 데이터 형식으로 초기

〔리스트 3〕 루프백 테스트 프로그램

```
PROGRAM RS232C_loopback;
VAR X,Y:CHAR;
    I:BYTE;
BEGIN
  FOR I:=$30 TO $39 DO
    BEGIN
      X:=CHR(I);
      WRITE(AUX,X);
      READ(AUX,Y);
      WRITELN(X,'     ',Y)
    END
END.
```

화 시킬 수 있으나 CP/M에는 이런 기능이 없으므로 프로그램을 직접 작성해야 한다. 이 경우 6850 Command 레지스터에 직접 값을 넣게 해서는 안된다. 6850을 직접 다루는 UR2.COM과 UP2.COM에서는 데이터를 입력 혹은 출력하기 전 CKINIT 서브루틴을 호출하여 6850이 초기화가 이미 된 상태인지 아닌지 판단한다.

판단의 기준은 포인터 VEC의 값이다. VEC에는 처음 0이 들어 있으나 6850이 초기화 되면 65H로 바뀐다. 따라서 초기화 루틴에서 6850 Command 레지스터에 넣을 값을 가지고 있는 부분을 원하는 값으로 변경시키고 VEC의 값을 0로 설정하면 된다. 이런 기능을 수행하는 프로그램이 **리스트 4**의 SETUP.PAS 파스칼 프로그램이다. 데이터 형식을 변경할 필요가 있을 때 이 프로그램을 실행시키고 화면에 나타난 메뉴중 원하는 것을 고르면 된다.

응용

애플용 RS-232C 인터페이스가 완성되고 테스트가 끝났으면 사용할 차례이다.

IBM-PC와 애플간의 화일전송실험을 해보자. 우선 두 컴퓨터간의 연결 케이블을 준비한다. 그림 7의 Null-모뎀 케이블을 직접 만들어야 한다. 사용선

이 RXD, TXD, SG의 3 가닥이므로, 심선이 2개 있고 망선으로 실드(Shield)된 케이블의 망선을 SG로 두 심선을 RXD와 TXD로 쓰면 된다. 이런 류의 케이블을 써야 에러가 없는 안전한 전송을 할 수 있다.

애플의 경우 인터페이스 카드에 달린 수코넥터에 케이블의 암코넥터를 끼우고 나서 인터페이스 카드를 슬롯(2번)에 설치하는 것이 편하다.

두 컴퓨터의 전원을 끈 상태에서 케이블을 설치하고 난후 전원을 켠다. 애플의 경우 사용 OS는 CP/M이다.

전송 프로그램은 IBM-PC는 MS-DOS 내장 명령어 Copy이고 애플은 CP/M 유틸리티 프로그램인 PIP를 사용한다. 우선 IBM-PC측의 통신형식을 애플측의 그것과 일치시킨다. 필자의 경우 애플측의 보오레이트 조정 딥스위치는 2400보오 위치에 놓았고, IBM-PC는 통신포트들중 Com2에 케이블을 접속시켰으므로 MS-DOS상에서

```
MODE COM2 : 2400,7,8,2
```

를 입력한다. 이제 실지 시험에 들어간다. 먼저 애플에서 IBM-PC쪽으로 텍스트 화일을 전송해 본다. 아스키코드로만 이루어진 텍스트 화일 외에 이진 화일을 전송하는 경우도 생길 수 있으나 두 기종간의 운영체제가 다르므로 이진 화일 자체가 상대방에게는 의미가 없다. 전송할 텍스트 화일명은 TEST.TXT이다. 먼저 수신측인 IBM-PC에서 다음 명령을 먼저 수행시킨다. 반드시 수신측이 먼저 준비상태에 있어야 한다.

```
COPY COM2 : TEXT.TXT
```

송신측 애플에서는 다음 명령을 실행시킴으로써 송신을 개시한다.

```
PIP PUN : = TEXT.TXT<E>
```

MS-DOS에서는 수신측이 오른쪽이 되지만 CP/M에서는 그 반대이므로 주의한다. CP/M에서 사용하는 표기 기준이 전산학의 표준 표기법이다. PIP 프로그램의 파라미터 [E]는 없어도 되지만 있으면 전송되는 내용이 애플 스크린에 표시되므로 편리하다. 전송작업이 끝났으면 TYPE 명령으로 제대로 전송되었는지 확인한다.

다음은 IBM-PC의 화일을 애플쪽으로 전송할 차례이다. 먼저 다음과 같이 수신측인 애플에서 준비상태에 들어간다.

```
PIP TEXT.TXT = RDR :
```

이때 절대 파라미터 [E]를 붙이면 안된다. 또한 안전을 위해 애플 디스크에 있던 TEST.TXT를 다른 이름으로 바꾸어 놓는 것이 좋을 것이다. IBM-PC에서는 다음과 같이 송신을 시작한다.

```
Copy text.txt /b com2
```

〔리스트 4〕 전송데이터 형식 SET UP 프로그램 SETUP. PAS

```
PROGRAM SETUP;
CONST ITEM:ARRAY[0..7] OF STRING[40]=
            ('7     EVEN    2',
             '7     ODD     2',
             '7     EVEN    1',
             '7     ODD     1',
             '8     NO      2',
             '8     NO      1',
             '8     EVEN    1',
             '8     ODD     1');
VAR I:BYTE;
    X,Y:BYTE;

BEGIN
  X:=MEM[$F2A8];
  Y:=(X-1) SHR 2;
  WRITELN;
  WRITELN('    **  DATA  PARITY  STOP  **');
  WRITELN;
  FOR I:=0 TO 7 DO
      BEGIN
        IF I=Y THEN LOWVIDEO;
        WRITE(I:2,' -     ',ITEM[I],' ');NORMVIDEO;
        WRITELN
      END;
  WRITELN;
  WRITE('     SELECT : ');READLN(X);
  WRITELN;
  IF X IN [0..7] THEN
     BEGIN
       Y:=(X SHL 2)+1;
       MEM[$F2A8]:=Y;
       MEM[$F2AD]:=0;
       WRITE('NEW SETUP ');LOWVIDEO;
       WRITE(' ',ITEM[X],' ');NORMVIDEO;
       WRITELN;WRITELN
     END
 END.
```

이때 /b 스위치를 붙이지 않으면 텍스트 화일의 끝을 나타내는 ^Z 문자가 전송되지 않으므로 애플측에서 수신을 종료할 수 없다.

참고로 애플 DOS상의 화일을 전송할 필요가 있을 때는 유틸리티 프로그램 APDOS를 이용해 CP/M 화일로 변환하면 된다.

이상과 같이 가장 기초적인 데이터 전송실험을 해보았다. 하드웨어도 간단하고 별도의 전송프로그램을 작성할 필요도 없으므로 누구나 해볼 수 있으리라 생각된다. 그러나 이와 같은 간단한 시스템일지라도 매우 효과적으로 잘 쓸 수 있다. 필자는 다음과 같이 활용하고 있다.

프린터는 IBM-PC에 연결되어 있고 롬라이터는 애플전용품이다. 애플에서 작성한 프로그램을 프린터로 인쇄할 때 번거롭게 무거운 프린터를 들고 왔다갔다 하지 않고 인쇄할 프로그램을 프린터가 연결되어 있는 IBM-PC로 전송한후 IBM-PC쪽에서 인쇄한다. 마이크로프로세서 6809 기계어 프로그램을 롬에 써넣을 때는, CP/M용 6809 크로스어셈블러를 못구했으므로, MS-DOS용 6809 크로스어셈블러에서 HEX 화일을 만든후 애플로 전송하여 롬에 써넣는다. 표 4에 있는 애플 화면출력·키보드입력 엔트리포인트를 RS-232C 입출력 포인트로 변경한후 IBM-PC에서는 통신프로그램인 크로스토크를 실행시켜 멋진 RGB화면으로 CP/M을 운용하기도 한다(IBM-PC를 애플의 터미널로 쓰는 예).

요즘 '데이터통신'이란 말을 접할 수 있게 되었고 실지로 데이터통신의 수요가 늘고 있다. 개인용 컴퓨터의 가치를 극대화하는 것이 데이터통신임을 생각할 때 퍽 반가운 일이다. 본고에서는 통신에 관한 기본적인 내용만을 서술했지만 이 내용들만 완전히 자기 것으로 소화한다면 훨씬 더 수준높은 데이터통신의 진수를 어렵지 않게 맛볼 수 있을 것이다.

본고에서는 또한 애플컴퓨터를 통신에 활용하는 방안을 제시하였다. 요즘 8비트 컴퓨터가 외면당하고 있지만 데이터통신이란 것이 고속연산이나 대용량 메모리 혹은 정교한 그래픽 기능을 요구하는 것이 아닌이상 8비트 기종도 충분히 데이터통신에 사용될 수 있다. 미국의 경우 애플용 모뎀과 통신소프트웨어들이 널리 보급되어 있다.

필요한 내용을 상세히 전하기 위해 나름대로 애썼지만 하드웨어나 기초이론에 문외한인 독자에게는 어려운 글이 된것 같아 죄송스럽다. 보잘것 없는 글이지만 독자여러분들께 도움이 되기를 바란다. PC

〈참고문헌〉

1. RS-232C 인터페이스의 사용법. 가남사 편집부.
2. Practical Data Communications. Fred Jennings 저.
3. Basic Micro processor and the 6800. Ron Bishop 저.
4. Micro Softcard Installation & Operation Mannual.
5. MS-DOS 핸드북. 정보시대.

DOS의 기능호출

FCB나 화일 핸들을 조작하고 또는 어떤 응용프로그램을 실행시킬때 시스템 내부에서 벌어지고 있을 각종 정보의 흐름을 이해하는 것은 시스템을 체계적으로 이해하고 분석할 수 있는 좋은 기반을 마련해 준다.

시스템 부팅

컴퓨터의 스위치를 켜면 몇초동안 특정한 화면 메시지가 나오고 도스프롬프트가 나타나면서 사용자와의 인터페이스를 마련해 놓는다. 흔하게 지나치는 일이지만 이 짧은 시간동안 시스템 내부에서는 실로 복잡한 일들이 진행되고 있다. CPU는 설계시 설정된 상태로 자기 자신을 초기화시키고 ROM속의 코드를 운영하기 시작한다. ROM은 일련의 하드웨어적인 초기화(메모리 테스트 등)를 수행한 뒤 부트 로더(Boot-loader) 프로그램을 로드하고 그쪽으로 제어를 넘겨준다.

부트 로더는 디스크로부터 IO.SYS나 IBMBIO.COM과 같은 프로그램만을 로드하는 기능을 갖는다. 이들 프로그램은 부트된 디스크의 루트디렉토리에서도 첫번째 엔트리, 즉 물리적으로 화일이 할당되는 공간에서 처음 부분을 점유하고 있게 된다. 이러한 사실에서도 엿볼 수 있듯이 부트 로더는 프로그램에 대한 디렉토리, 서브디렉토리, 그리고 패스 등의 정보까지 인식할 정도로 섬세하지가 못하다.

IO.SYS(또는 IBMBIO.COM)는 컴퓨터의 초기화를 끝내고 MS-DOS(구체적으로 서술하면 MSDOS.SYS 또는 IBMDOS.COM)를 로드한다. 이렇게 하여 오퍼레이팅 시스템이 올바르게 들어와서야 비로소 컴퓨터는 디스크화일 및 디렉토리를 찾을 수 있게 되고 메모리를 운영하며 시간과 요일 등의 트랙을 유지할 수 있게 되고 또한 명실공히 칩과 디바이스들의 조합체를 떠난 컴퓨터로서의 의미를 갖게 된다.

다음에, MS-DOS는 CONFIG.SYS라는 화일을 읽어 들여서(만약 이 화일이 있다면) 그속에 포함된 명령어들을 수행한다. 통상 이들 명령어들은 시스템 내부의 MS-DOS테이블을 변경시키거나 디바이스 드라이버를 로드하고 초기화 시키기도 한다. 그리고 CONFIG.SYS속의 명령들은 컴퓨터가 사용자와의 인터페이스를 형성하기 전, 즉 부팅되면서 해석되기 때문에 부팅과정에서는 CONFIG.SYS를 지울 수 없다.

시스템이 CONFIG.SYS의 실행을 끝내면 COMMAND.COM(또는 CONFIG.SYS내에 지정된 다른 유저용 셀)을 로드하여 제어를 그쪽으로 넘기게 된다. 제어를 넘겨받은 COMMAND.COM은 부팅 디스크의 루트디렉토리에서 AUTOEXEC.BAT라는 화일을 찾기 시작한다.

만일 이 화일이 존재하면 COMMAND.COM은 AUTOEXEC.BAT속의 각 명령 라인을 로드하여 실행하고 이 화일이 없으면 화면을 통하여 요일 및 시간을 묻는 메시지를 출력한다. 이제 컴퓨터는 사용자의 입력을 받아들일 준비가 되었으며 COMMAND.COM은 화면에 A> 또는 C> 프롬프트를 만들어 내고 사용자가 명령을 내리기를 기다린다. 지금까지는 컴퓨터 전원스위치를 켰을 때 내부에서 순간적으로 수행되는 과정을 살펴본 것이다. 물론 이와 같은 과정은 재부트(Ctrl-Alt-Del)시에도 일어난다.

COMMAND.COM의 주 기능

COMMAND.COM은 MS-DOS의 화면-인터페이스 기능을 제공하고 있다. 그래서 어떤 종류의 명령, 또는 명령 처리방식이 마음에 들지 않으면 COMMAND.COM을 수정하여 변경시킬 수도 있다. 물론 기존의 내용을 손상시키지 않고 새로운 MS-DOS 명령을 추가하여 COMMAND.COM을 확장시킬 수도 있다.

정확히 말하자면 COMMAND.COM은 정상적인 .COM 프로그램은 아니지만 MS-DOS의 대부분의 규칙들을 따르고 있다. 그 주요 기능을 살펴보면, 명령을 대기하고 있다가 어떤 명령이 입력되면 그것을 해석하고 다음, MS-DOS에 적절한 호출을 하여 명령을 수행하는데 있다.

예를 들어 A> 프롬프트에 어떤 명령을 입력하면 COMMAND.COM은 먼저 DIR 또는 COPY 등과 같은 내부명령어인지를 조사하고 만일 그렇다면 DOS인터럽트 21(INT 21H) 서비스에 호출을 통하여 그 명령을 수행한다.

만일 COMMAND.COM이 그의 내부 테이블에서 입력된 명령을 찾지 못하면 입력한 워드에 .COM을 추가하여 현재의

디스크 디렉토리에 그 이름과 똑같은 화일이 있는지 조사하고 이 때도 실패하면 .EXE 및 .BAT확장을 시도하여 다시 조사한다.

만일 계속하여 매칭이 이루어지지 않으면 "Bad command or file name"과 같은 메시지를 디스플레이하고, 또는 최초 매칭이 .BAT로 끝나는 화일이라면 COMMAND.COM은 그 배치화일 모드로 들어가서 그곳의 명령들을 수행한다.

호출 정보

COMMAND.COM과 그 주변에서 일어나는 정보를 조사하기 위해 COMMAND.COM화일을 파헤쳐 버리는 불상사는 가급적 없어야 한다. 다만 우리는 MS-DOS에 COMMAND.COM이 무엇을 호출하고 있는지에 대해서만 알고 싶은 것이다.

만약 메모리에 상주할 수 있는 디버깅 프로그램이 갖추어져 있다면 그것을 로드하고 DOS서비스 요구기능인 INT 21H의 벡터 어드레스에 영구절점(Permanent Break Point)을 세트할 수 있다.

벡터 어드레스를 찾아내는 손쉬운 방법은 0:84HEX(84HEX=21H*4) 번지에서 시작하는 4바이트를 읽어 들인 다음 그것을 역순으로 놓는 것이다.

예를 들면 어떤 시스템에서 0:84에서 시작하는 4바이트가 7C, 05, AF, 그리고 1A라고 하면, 이러한 바이트들을 역순으로 놓는다는 것은 INT 21H 벡터가 메모리 번지 1AAF:057C에서 시작한다는 것을 보여준다. 리스트 1의 베이직 프로그램은 이와 같이 바이트값을 재정렬시켜 준다.

MS-DOS에 대한 COMMAND.COM의 호출을 찾아보기 위해 디버깅프로그램을 이용할 때 생기는 문제는 모든 호출을 프로그램 실행도중에 밖에 관찰할 수 없다는 점이다. 즉 MS-DOS에 관한 기술적인 매뉴얼을 전전하며 INT 21H 요구들의 리스트를 구성하는 것보다 프로그램을 통하여 호출에 대한 리스트를 얻는 것이 더 효율적이다.

이러한 과정에서 리스트 2의 서비스 프로그램은 편리한 기능을 제공하게 된다. 이 프로그램은 INT 21H로 만들어진 호출을 모아서 그들 각 요구에 해당하는 서비스 번호와 메모리 테이블을 저장한다.

〔리스트 1〕 INT 21H의 인터럽트 벡터 번지를 구하는 베이직 프로그램

```
1 '  Simple Basic Program to find
2 '  the interrupt vector for Int 21h
3 '
100 DIM BYTES(4)
110 DEF SEG = 0
120 FOR J = &H84 TO &H87
130     BYTES(J-&H83) = PEEK(J)
140 NEXT J
150 PRINT "Int 21h vector is ";
160 FOR J = 4 TO 1 STEP -1
170     A$ = HEX$(BYTES(J))
180     WHILE LEN(A$) < 2
190       A$ = "0" + A$
200     WEND
210     PRINT A$;
220     IF J = 3 THEN PRINT ":";
230 NEXT J
```

〔리스트 2〕 DOS 서비스 요구를 찾기 위한 메모리상주 프로그램

```
code     segment
         assume cs:code, ds:nothing, es:nothing

         org     100h                      ;Normal COM program start
start:   jmp    install

new_21:  pushf                             ;Save Caller's flags
                                           ; (probably not necessary)
         cmp     cs:list_ptr,offset message   ;Is the message full?
         jae     continue                  ;Yes -- go

         push    di                        ;Else save caller's DI
         mov     di,cs:list_ptr            ;Get the list pointer
         mov     byte ptr cs:[di],ah       ;Save this service number
         inc     cs:list_ptr               ;Point to next position
         pop     di                        ;Recover caller's register

continue:
         popf                              ;Restore caller's flags
         jmp     cs:[old_21]               ;And go

; Memory-resident data area:

list_ptr   dw   offset list                ;Point to next storage area
old_21     dd     ?                        ;Previous Int 21h address
           db   'Here is the list -->'     ;Helps us find list in memory
list       db   400h  dup (-1)             ;Room for 1024 calls

;Transient data area
```

```
message db    'Service installed at '
addr    db    9 dup (' ')
        db    13,10,'$'

hex_tbl db    '0123456789ABCDEF'

; Installation code

        assume   ds:code,es:code    ;Here we can use DS and ES

install:
        mov    di,offset addr      ;Ready to build installation
        mov    ax,cs               ;  address
        call   ax_to_hex           ;Convert segment to hex ASCII
        mov    al,':'
        stosb
        mov    ax,offset old_21+4  ;Now get offset address
        call   ax_to_hex           ;Convert it

        mov    dx,offset message   ;Display message
        mov    ah,09h              ;  which old Int 21h is still
        int    21h                 ;  installed

        mov    ax,3521h            ;Get vector of Int 21h
        int    21h
        mov    word ptr old_21,bx  ;Save current Int 21h
        mov    word ptr old_21+2,es ;  segment & offset
        mov    dx,offset new_21    ;Now install our routine
        mov    ax,2521h            ;   as Int 21h
        int    21h

        mov    dx,offset  message ;First trancient address
        int    27h                ;Terminate but stay resident

 ax_to_hex proc   near
        mov    bx,offset hex_tbl  ;Table of hex digits

        xchg   ah,al              ;Start with high byte
        push   ax                 ;Save it
        mov    cl,4
        shr    al,cl              ;High nibble to low
        xlat   hex_tbl            ;Change to hex-ASCII
        stosb                     ;Place in message
        pop    ax
        and    al,0fh             ;Mask off high nibble
        xlat   hex_tbl
        stosb                     ;Save that digit

        xchg   ah,al              ;Now start again with low byte
        push   ax                 ;Save it
        mov    cl,4
        shr    al,cl              ;High nibble to low
```

그리고 ORG 100H로 .COM형식을 취해 메모리에 상주됨을 의미하고 있으며 리스트의 중간부분에 있는 install 루틴으로 분기되면서 시작되고 있다.

결과적으로 이 프로그램은 자신의 내부 테이블을 16진수로 변환하여 이것을 화면으로 출력시켜 주게 된다.

그밖에 이 프로그램은 원래의 INT 21H 벡터의 번지를 기억하기 위해 DOS 기능 호출을 이용하며 현재의 INT 21H인 new_21 루틴의 번지를 초기화시키는데도 역시 DOS기능을 호출하고 있다. 그래서 프로그램을 종료하고 INT 27H를 통해 빠져 나오지만 메모리속에 자신의 상주영역은 여전히 남긴다.

일단 이 프로그램의 초기화가 이루어지면 그후의 어떠한 프로그램이든지 INT 21H를 통한 호출은 리스트 2의 new_21 루틴을 거치게 된다. 리스트 2의 처음 부분에 이 루틴이 있는데 바로 여기서 각 서비스 요구들이 메모리 테이블에 저장된다. 또 여기서 list라는 테이블과 그속에 비어있는 영역을 가리키는 list_ptr이라는 포인터 변수를 유지하고 있어서 이 루틴이 불려질 때마다 테이블내의 빈 영역을 확인하기 위해 list_ptr의 값을 조사한다. 그래서 이 테이블에 잔여공간이 있으면 서비스 요구 레지스터인 AH내에 현재 값을 저장한 후 포인터를 수정한다.

리스트 2는 간단한 프로그램이기는 하나 우리가 필요로 하는 정보를 훌륭하게 출력시켜 준다. 그리고 한가지 유의할 점은 이 프로그램을 실행하기에 앞서 이미 다른 종류의 메모리 상주프로그램이 실행되었는지를 체크해야 한다는 점이다.

초기상태라면 MS-DOS 프롬프트상에서 SERVICE를 입력하여 실행시킨다. 화면에 초기화 메시지와 함께 어드레스 번호가 나타나는데 이를 기록해 둔다(한 예로 제우스컴퓨터의 EXP/10 AT기종인 경우 0DCE:0128가 나옴) 이 상태에서 즉시 도스 프롬프트가 나오면 DEBUG를 입력한다. DEBUG

```
        xlat   hex_tbl          ;Change to hex-ASCII
        stosb                   ;Place in message
        pop    ax
        and    al,0fh           ;Mask off high nibble
        xlat   hex_tbl
        stosb                   ;Save that digit
        ret
ax_to_hex  endp

code   ends
       end    start
```

COMMAND.COM에 의해 만들어진 호출들의 한 예를 표 1에 보였는데 여기서 보면 어떤 프로그램을 실행시키기 위해서는 DOS 서비스기능 4BH를 이용하고 있음을 보게 된다.

화일핸들과 화일제어블록

INT 21H 서비스기능 호출들의 리스트를 자세히 살펴보면 화일을 조작하는 방법에는 두가지가 있음을 알게 된다.

〔표 1〕 SERVICE.COM의 실행과 DEBUG가 로드된 사이에 발생한 DOS 서비스 요구들의 예

4D	Service.COM으로 부터 코드값을 입력
48	(두번) Service.COM의 메모리와 Command.COM의 메모리 제어를 해제
25	(세번) Command.COM의 제어 벡터 설정
3E	(15번) 모든 화일의 Close.
37	·
38	·
40	화일 쓰기
19	(두번) 현행 드라이브
40	(두번)
5D	(두번) 공개되지 않은 DOS Call
0A	입력 버퍼
29	(세번) 화일이름 체크
1A	DTA 설정
47	현행 드라이브와 패스명
4E	일치되는 화일이름 찾기
47	패스 내부 보기
4E	·
49	새로운 프로그램을 위한 메모리 할당
4B	프로그램 실행

프롬프트(—)상에서 D(dump) 명령을 이용하여 방금 기록한 어드레스값을 입력한다.

만일 모든 과정이 순조롭게 진행되었다면 DEBUG는 메모리속의 한 테이블을 보여 줄 것이다. 테이블의 한 부분은 무질서하게 보이는 바이트값들로 채워져 있고 나머지는 0FF로 채워져 있을 것이다. 이 리스트가 바로 SERVICE.COM이 INT 27H를 실행했을 때부터 DEBUG가 그 테이블을 화면에 출력한 때까지의 MS-DOS에 대한 COMMAND.COM과 DEBUG의 호출 일람표이다. 작업이 끝난 후 이 프로그램을 메모리로부터 제거하려면 시스템 자체를 재부트시켜야 한다(Ctrl-Alt-Del).

즉 화일을 열거나 읽어들이기, 저장하기, 화일닫기 등의 조작은 두가지 방법으로 할 수 있다는 말이다.

각 한쌍으로 구성된 서비스기능 번호에서 하위번호는 화일제어블록(FCB)을 이용하고 상위번호는 화일핸들(File Handle)을 이용하고 있다. FCB화일 서비스는 CP/M 오퍼레이팅 시스템과 호환성을 가지며 화일 핸들에 기초를 둔 서비스는 UNIX시스템 호출과 비슷하게 설계되어 있음을 알아 두자. 이들 두 방법의 상이한 점을 또한가지 든다면, FCB 호출을 이용하는 프로그램은 모든 버전의 MS-DOS에서 사용할 수 있지만 화일핸들에 기초를 둔 서비스기능은 MS-DOS 2.0과 그 이상의 버전에서 가능하다는 점이다.

프로그램이 화일핸들 서비스로 화일이름을 전달할 때 DS:DX레지스터는 화일이름을 가리키게 되고 화일이름은 또 ASCIIZ 형식(제로바이트로 끝나는)으로 기억되어 있어야 한다.

FCB를 이용하는 경우는 좀더 복잡해 진다. FCB는 '닫혀진 상태' 또는 '열린 상태'라든지 '정상 상태' 또는 '확장 상태'에 있게 되며, 모든 경우에 DS:DX는 프로그램이 MS-DOS를 호출할 때마다 FCB를 가리킨다. MS-DOS가 화일을 오픈할 때 FCB의 정보를 일부 수정하게 되는데 바로 이 정보들을 통하여 응용프로그램과 MS-DOS가 서로 교신하고 있다.

일반적으로 오픈되지 않은 FCB에서 최초의 바이트는 디폴트 드라이브를 지정하는 숫자 0이나 지정 드라이브(1=A, 2=B 등)를 나타내는 번호를 갖는다. 그리고 FCB에는 패스에 관한 정보가 없기 때문에 화일은 반드시 현재 액세스되는 드라이브의 디폴트 디렉토리에 있어야 한다. 이런 점들을 고려해 보면 화일핸들을 이용하는 것이 쉽다는 것을 알 수 있다.

리스트 2는 COMMAND.COM이 만들어 내는 호출들을 보여주고 있지만 각각의 호출때마다 전달되는 정보를 알려주고 있지 않다. 이와 같은 DOS 서비스기능 요구에 필요한 정보들을 알아내고 그 정보들을 기억하거나 또는 화면에 보

여주는 프로그램은 다소 복잡하게 된다. 즉 MS-DOS에 대한 어느 정도의 지식을 필요로 한다. 그러나 FCB와 Handle 화일의 오픈, 프로그램의 실행과 같은 세경우의 요구동안에 MS-DOS로 전달되는 정보에만 제한한다면 MS-DOS에 프로그램들이 어떻게 기능 요구를 행하고 있는지를 보이는 것은 그리 어렵지 않다. 리스트 3이 이와 같은 프로그램인데 install 루틴을 보면 자신의 어드레스를 나타내지 않는다는 점만 제외하고는 리스트 1과 비슷하게 되어 있다.

리스트중의 new_21 부분은 FCB와 Handle화일의 오픈, 프로그램 실행 등을 위하여 AH레지스터속에 들어있는 서비스 요구 번호를 체크하면서 시작된다.

만일 호출이 위에서 설명한 세 경우가 아닌 경우는 레지스터에 아무런 변화없이 본래의 INT 21H로 분기된다. 만일 원하는 요구가 위의 세경우 중의 하나와 일치한다면 프로그램상태는 리포트 레벨로 분기하고 DS와 DX레지스터 뿐만 아니라 스택 세그먼트 및 포인터를 저장하고 내부 스택의 모든 CPU레지스터들을 저장한다. 그리고 나서 현재의 코드 세그먼트를 가리키도록 DS와 ES를 세트시킨다.

리스트 3은 출력할 세 종류의 메시지들 중의 하나를 선택하기 위해 AH내의 서비스 요구 번호를 참조한다. 이들 메시지들은 세가지 가능한 서비스 요구의 이름이다. 서비스 요구 번호는 다시 FCB를 기초로 요구가 생성되는지, 아니면 화일핸들에 기초해서 요구가 생성되는 지를 정하기 위해 참조된다(서비스 4B는 기술적으로 화일핸들에 기초를 둔 요구이다).

프로그램을 실행시키면 화일의 오픈, 또는 실행요구로 MS-DOS에 전달되는 화일이름을 볼 수 있다. 그리고 실행하고자 하는 응용프로그램이 FCB에 기초한 서비스 기능을 이용하는지 아니면 화일핸들에 기초한 서비스 기능을 이용하고 있는지를 알 수 있을 것이다.

〔리스트 3〕 MS-DOS 서비스 요구들을 제어하는 프로그램

```
 ;
 ; Register save/restore macros
 ;
 RPUSH  macro  list
            ;;List of registers to save
     irp   dummy,<list>  ;;Turn registers
     push  dummy         ;;Push each in turn
     endm                ;;   End of IRP
     endm                ;;   End of RPUSH

 RPOP  macro list
           ;;List of registers to retreve
     irp dummy,<list>    ;;Turn registers
     pop  dummy          ;;Pop each in turn
     endm                ;;    End of IRP
     endm                ;;    End of RPOP

 code  segment
       assume  cs:code, ds:code, es:code

        org   100h
 start: jmp   install

 new_21: pushf
      cmp   ah,0fh        ;Is call FCB Open?
      je    report        ;Yes -- report it
      cmp   ah,3dh        ;Is call Handle Open
      je    report        ;Yes -- report it
      cmp   ah,4Bh        ;Is call EXEC ?
      je    report        ;Yes -- report it
 cont:  popf              ;Else restore stack
      jmp   cs:[old_21] ;And go

 report:
   mov  cs:stk_seg,ss  ;Save caller's stack
   mov  cs:stk_off,sp  ; in our data area
   mov  cs:data_seg,ds ;We'll need DS and DX
   mov  cs:call_ptr,dx ; later
   push  cs            ;Move to local
   pop  ss             ; stack
   mov  sp,offset stack
   sti                 ;Now start interrupts

   RPUSH <ax,bx,cx,dx,si,di,ds,es>
                    ;Save everything

   mov  bx,cs       ;Make all segment
   mov  ds,bx       ;  registers local
   mov  es,bx
   push  ax        ;We'll need it again later
   mov   dx,offset fcb_o$
                    ;Ready to print FCB Open
   cmp  ah,0fh      ;Is that right?
   jz   prntl       ;Yes -- go
   mov  dx,offset hndl_o$
                    ;Then print Handle Open
   cmp  ah,3dh      ;Is that right?
   jz   prntl       ;Yes -- go
   mov  dx,offset exec$
                    ;Else it is an EXEC call
 prntl: mov  ah,09h
    int  21h        ;Print 1st message

    pop  ax    ;Get service request again
```

```
    push ds   ;Save our data segment
    mov  ds,data_seg   ;Get caller's DS
    mov  si,cs:call_ptr
              ;DS:SI ==> file name or FCB
    mov  di,offset cs:file_nam
              ;Pointer to file name buffer

    cmp  ah,0fh   ;FCB open?
    jne  hndl1 ;No--use handle-name logic

    lodsb      ;Get first byte of FCB
    cmp  al,0ffh  ;An extended FCB?
    jnz  fcb2     ;No -- go
    add  si,6     ;Else skip extension
    lodsb         ;Get drive number

fcb2: or   al,al   ;Is it default drive?
    jz   fcb3     ;Yes -- don't report it
    add  al,'A'-1 ;Else change to drive letter
    stosb         ;Store in our buffer
    mov  al,':'   ;Insert drive colon
    stosb

fcb3:  mov  cx,8   ;8-byte file name
    rep  movsb    ;Move file name to our buffer
    mov  al,'.'   ;Now the separator
    stosb
    mov  cx,3     ;Next comes the file extent
    rep  movsb
    jmp  short hndl2
              ;Add carriage return/line feed

hndl1: lodsb     ;Get a byte
    or   al,al  ;At the end?
    jz   short hndl2  ;Yes -- go
    stosb        ;Else move to our buffer
    jmp  hndl1  ;And repeat

hndl2: mov  al,13  ;Carriage return
    stosb
    mov  al,10    ;Line feed
    stosb
    mov  al,'$'   ;String terminator
    stosb

    pop  ds       ;Bring DS back here
    mov  dx,offset file_nam
              ;Now print the file name
    mov  ah,9
    int  21h

    mov  ah,08h   ;Wait for a keystroke
    int  21h
    RPOP <es,ds,di,si,dx,cx,bx,ax>
              ;Restore registers
    cli           ;Interrupt off
    mov  ss,cs:stk_seg ;to switch stack
    mov  sp,cs:stk_off
    jmp  cont     ;And leave

;
; Memory-resident data
;
old_21  dd  ?      ;Original Int 21h
stk_seg dw  ?      ;Save stack segment
stk_off dw  ?      ;   and stack offset
data_seg dw  ?     ;Save data segment
call_ptr dw  ?     ; and pointer to call info
fcb_o$   db  13,10,"FCB Open: $"
hndl_o$  db  13,10,"Handle Open: $"
exec$    db  13,10,"EXECute: $"
file_nam db  66 dup (' ')
              ;Space for file/path name

    EVEN          ;Put stack on even boundary
    dw  40h  dup (?)
stack  dw  ?      ;Our stack goes here

;
;  Transient data
;
inst$ db "Name catcher is installed",13,10,'$'

install:
    mov  dx,offset inst$ ;Display message
    mov  ah,09h  ;while old Int 21h is still
    int  21h     ;  installed

    mov  ax,3521h  ;Get vector of Int 21h
    int  21h
    mov  word ptr old_21,bx
              ;Save current Int 21h
    mov  word ptr old_21+2,es
              ;  segment & offset

    mov  dx,offset new_21
              ;Now install our routine
    mov  ax,2521h  ;  as Int 21h
    int  21h      ;We're installed!

    mov  dx,offset inst$
              ;First transient address
    int  27h      ;Terminate but stay resident

 code  ends
       end  start
```

디스크드라이브의 응급처치

컴퓨터를 사용할 때 가장 빈번히 발생하는 에러는 데이터 에러이다. 이러한 에러의 근본원인은 디스크드라이브의 에러이다. 따라서 디스크드라이브의 내부를 잘 알고 응급조치를 취할 수 있다면 이러한 에러를 미연에 방지할 수 있을 것이다.

컴퓨터 사용중 데이터 에러의 가장 주된 원인은 디스크 드라이브의 결함이다. 이러한 상황에서 모든 이용자가 디스크를 부트 업(Boot up)하면 데이터 에러를 얻게 되거나 에러 또는 리턴을 치므로써 어떤 다른 거부 메세지를 발견하게 된다.

디스크 결함의 원인

어느 디스크드라이브도 기계적인 부분과 전자적인 부분으로 이루어져 있다. 디스크드라이브의 주요 기계적 성분은 새시(Chassis), 도어 메카니즘(Door Mechanism), 허브(Hub), 드라이브모터(Drive Motor), 스핀들(Spindle), 헤드-포지셔닝모터(Head-Positioning Motor) 및 헤드 어셈블리(Head Assembly), 로직 보드(Logic Board), 서보 보드(Servo Board) 등이다(그림 1, 2 참조). 비록 일부분의 제품에서는 서보회로가 로직보드에 포함되어 있지만, 대부분의 경우 로직보드와 서보보드는 분리되어 있다.

드라이브의 전면에 부착된 플라스틱 케이스는 도어 어셈블리와 도어에 접속된 디스크 리테이닝 허브(Disk Retaining Hub)를 안전하게 보호한다.

드라이브에 있는 2개의 모터중 하나인 드라이브모터는 스핀들을 회전시킴으로서 디스크 자체를 회전하도록 한다. 스핀들은 디스크의 큰 중심 홀에 위치한 허브에 꼭맞게 되어 있다. 디스크드라이브를 닫으면 디스크는 허브 클램프(Hub Clamp)에 의해 눌려 있는 상태로 된다(그림 3 참조).

4가지의 주된 문제 즉, 모터의 타버림, 모터 베어링 마모, 드라이브 벨트 결함, 부정확한 스핀율은 주로 드라이브 모터와 스핀들에 관련된다.

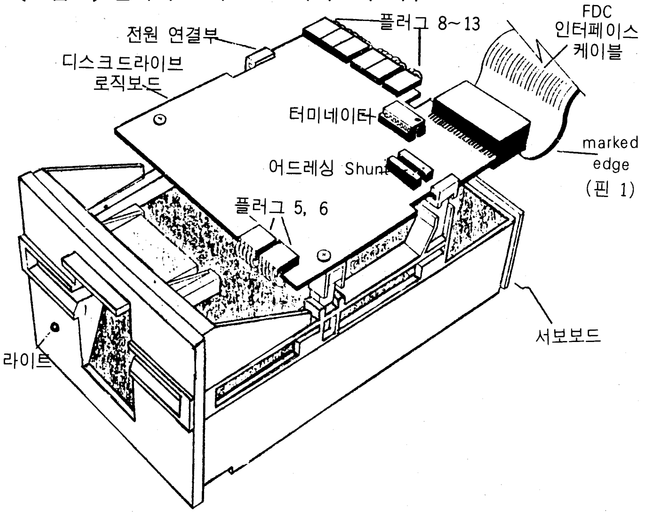
때때로 스핀들 베어링이 마모되어 스핀들의 회전을 억제시키기도 한다. 만일 전기적인 문제에 의해 스핀율(Spin Rate) 편차가 발생하면, 서보(Servo)단자 즉, 가변 저항을 트위킹(Tweaking)함으로서 정확하게 조정할 수 있다. 그러나 모터의 동작이 점점 나빠지고 있다면 이러한 트위킹은 단지 플리팅(Fleeting)의 안전을 제공할 뿐이다. 또한, 드라이브 벨트의 미끄러짐은 불규칙한 스핀율을 발생시킨다. 그리고 절단된 벨트는 물론 스핀율을 록-솔리드 제로(Rock-Solid Zero)로 떨어 뜨린다.

헤드 포지셔닝모터는 신속하고 정확하게 헤드를 디스크의 리드/라이트 슬롯 위에서 이리저리 움직이게 한다. 여기서 고장의 주요 원인은 헤드의 미스얼라인먼트(Misalignment)이다. 이 미스얼라인먼트는 디스크드라이브의 난폭한 취급, 마모, 과열 등으로 기인한다.

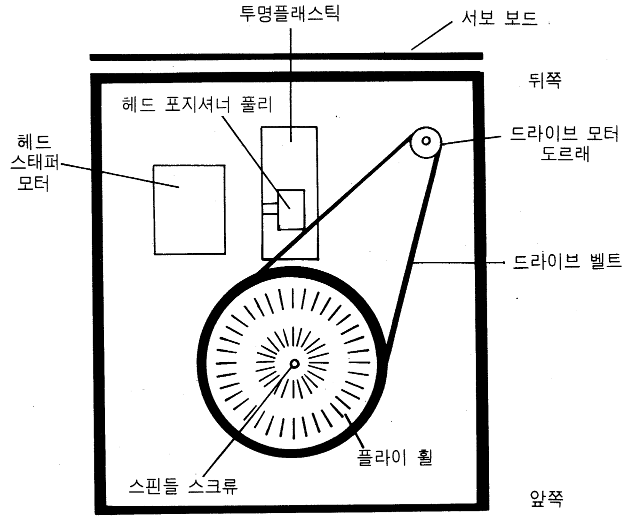
일반적으로 드라이브는 최상의 주위환경에서는 정상동작을 한다. 특히 불결한 주위환경에서 컴퓨터를 동작시킨다면 드라이브는 불규칙한 동작을 하게 될 것이다. 이때 드라이브를 직접 고치려고 하지 말아야 한다. 헤드 배치(Head Alignment)는 적당한 기술을 갖고 있는 전문가에게 맡겨야 한다.

컴퓨터의 작업수행은 DOS에서 전달되는 명령들에 따르게 된다.

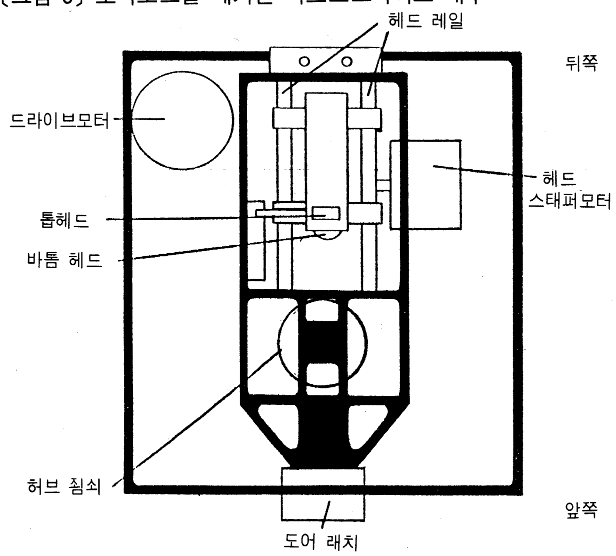
〔그림 1〕 풀사이즈 디스크드라이브의 내부



〔그림 2〕 디스크드라이브의 밑면



〔그림 3〕 로직보드를 제거한 디스크드라이브 내부



그러나, 그와 같은 명령들의 실제 수행은 로직보드에 접속된 플로피디스크 콘트롤러보드(Floppy Disk Controller Board; FDCB)—디스크 드라이브상에 있는 회로들 그 자체—에 의해 수행된다(그림4 참조).

IBM에서 디스크 어댑터보드로 알려진 FDCB는 컴퓨터의 확장 슬롯들 중의 하나에 장치된다. FDCB는 디스크드라이브의 오동작의 원인이 될 수 있지만, 이 FDCB의 양호한 상태를 확보하는 유일한 실제 방법은 제어보드를 성능이 우수한 것으로 교체하는 것이다. 플로피디스크 제어보드에 대한 고장수리는 일반인이 할 수 있는 일이 아니다.

명령들의 체인(Chain)

컴퓨터에 화일을 리드하라고 지시하게 되면, 적당한 신호들이 맨 먼저 FDC에 보내지고 그 다음에는 디스크드라이브 로직보드에 보내져서, 디스크드라이브를 동작시키며, 리드/라이트를 로드하며 헤드 스테퍼(Head Stepper)를 디스크의 표면에 횡으로 앞뒤로 움직이게 한다. 전기적인 동작과 전기기계적인 동작이 순조롭게 되도록 주의깊게 조정해야 시스템이 안전하게 동작하게 된다. 아직도 결함이 나타나면 IC의 결함, 그렇지 않으면 캐퍼시터나 트랜지스터의 개방이나 단락에 의한 것이다. 그리고 이러한 결함에 대한 보수도 역시 행해져야 한다.

서보보드(Servo Board)에 문제가 발생하면 회로보드(Circuit Board)를 교체하는 것이 이상적이다. 또는 서보보드가 작업을 수행못하면 이미 언급한 스핀들(Spindle)-속도-조정 제어로 스핀율(Spin Rate)을 조정해 본다.

일반적으로 디스크드라이브의 동작은 7개의 기능 그룹으로 세분할 수 있다. 이들은 다음과 같이 나눈다.

1. 인덱스 펄스 쉐이퍼(Index-Pulse Shaper)

이것은 LED 즉, 포토트랜지스터로 구성되어 있다. 그리고 펄스 쉐이핑회로들. 펄스 쉐이퍼는 LED상을 지나는 디스크의 인덱스 홀에 의해 발생되는 빛을 포획(Interrupt)하여 표준 디지탈 데이터 스트림(Digital Data Stream)을 형성한다. LED의 타버림(Burn Out)과 포토트랜지스터의 결함은 치명적으로 성능에 영향을 미친다.

2. 라이트 보호센서(Write- Protect Sensor)

이것은 디스크상의 라이트 보호 노치(Notch)가 언제 씌워지는지, 마이크로스위치가 언제 닫히는지, 그리고 라이트기능이 언제 정지하는지를 감지한다.

3. 트랙 00 센서

트랙 00의 위치는 헤드 스테퍼(Head Stepper)의 홈 베이스(Hom Base)이다. 만일 드라이브가 얼라인먼트(Alignment)를 벗어나면, 디스크의 트랙 00을 리드하지 못하여 깨지게 된다.

4. 스핀들 드라이브 제어

5. 캐리지 위치 제어

6. 라이트/이레이저(Write/Erase) 제어

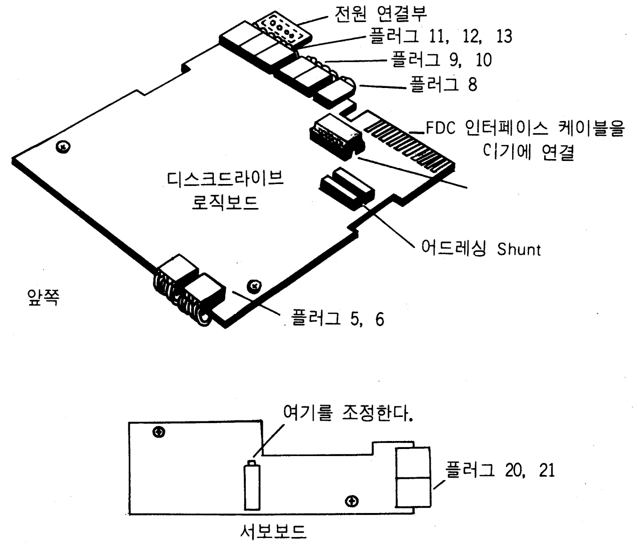
7. 리드 엠프리파이어(Read Amplifier) 및 디지타이저(Digitizer)

이 그룹은 로직보드의 전자적인 주요부품을 포함한다.

다른 문제들

플로피디스크 드라이브는 싱글 또는 더블 사이드 중의 하나이다. 싱글사이드 디스크드라이브는 단지 하나의 리드/라이트 헤드를 갖는다. 이 드라이브는 단밀도 모드에서 사용될 때 가장 안전한 형이다. 더블사이드 디스크드라이브는 두개의 헤드를 가지며 그중 두번째 것은 싱글사이드 드라이브

〔그림 4〕 디스크드라이브 로직보드와 서보보드



가 간단한 압력 패드(Pressure Pad)를 갖는 곳에 위치한다.

두개의 헤드를 갖는 드라이브는 하나의 헤드를 갖는 드라이브보다도 특히 더블 및 쿼드러플(Quadruple)—밀도 모드에서 미스얼라인먼트(Misalignment)에 민감하다. 구형 IBM 들은 싱글—사이드 드라이브(배밀도로 라이트 한다)를 갖는 반면, 신형 컴퓨터들은 더블사이드 드라이브(또는 배밀도)를 장치하고 있다. 레코딩 밀도는 FDC 보드의 제어하에 있으며, 이것은 어떤 디스크드라이브도 배밀도 정보를 리드하고 라이트할 수 있다는 의미이다. 확장된 밀도는 또한 이러한 드라이브에 의해 압축된다. 그러나 쿼드 덴시티(Quad Density)가 사용되어야 하는 곳에 일반적으로 "정규" 40-트랙(48트랙/인치) 드라이브 대신에 80-트랙(96트랙/인치)을 규정한다. 80-트랙 폭은 너무 좁아서 스펙으로 부터 약간의 편차(Deviation) 조차도 시스템을 불안하게 한다는 것을 유의하라.

FDC 보드는 표준 34핀 케이블을 통해 디스크드라이브와 교신한다. 어쨌든, 디스크드라이브 문제는 케이블이나 코넥터에 기인한다.

컴퓨터에서 디스크의 포매팅 프로세스는 표준 테이프 레코딩 작업과 유사하다. 드라이브가 기록하는 정보는 디지탈이지만, 테이프 레코딩에서 마그네틱 매질에 기록되는 아날로그 정보에 반대되는 것으로, 1과 0들로서 시스템이 해독할 수 있는 마그네틱 패턴이 존재한다는 의미이다.

포매팅 프로시듀어는 운영 시스템이 화일을 저장하고 복원하는데 사용하는 한 세트의 디지탈 피젼 홀(Digital Pigeon Hole)을 세팅함으로서 수행된다. 소형 패키지에 기록된 화일들은 서로 인접할 필요는 없으므로 포매팅은 운영 시스템이 체계화한 모든 것을 그대로 유지시키는데 도움을 주는 포인터(Pointer)나 "로드 사인(Road Sign)"을 제공한다.

현상과 조치방법

디스크드라이브를 완벽하게 고장 보수를 했다 하더라도 모든 것이 기대한만큼 기능을 발휘하지 못할 때가 많다. 여기에 고장 내용, 가능한 원인, 수리기술을 수록하였다.

●부트명령을 주었을 때 드라이브가 반응이 없다면 비지 라이트(Busy Light)가 동작이 안된다.

—디스크드라이브 케이블이 안전하게 플러그에 꽂혀 있지 않을 수 있다.

—결함이 있는 케이블(하나 이상의 절단된 컨덕터(Conductor))이 있을 수 있다. 이것을 교체하라.

—파워서플라이 케이블이 디스크 드라이브상의 파워 소켓에 연결되어 있지 않을 수 있다.

—FDC 보드 또는 디스크드라이브 로직보드상에 문제가 발생한 것일 수도 있다.

●부트명령을 주었을 때 디스크 비지 라이트는 켜져 있지만 디스크는 부트가 되지 않는다.

—디스크상에 시스템이 없다.

—디스크의 결함

—스핀율(Spin Rate)이 스펙(Spec)을 벗어남. 그것을 조사한 다음 조정하라.

—드라이브 벨트가 끊어졌거나 풀레이(Pulley)에서 벗겨짐. 그것을 교체하라.

●드라이브 모터의 정지. 거기에 윤활유를 쳐라.

●헤드-스테퍼모터(Head-Stepper Motor)의 정지. 그것을 닦고 윤활유를 쳐라.

●드라이브모터 또는 스테퍼 모터의 타버림. 드라이브를 수리해야 한다.

●터미네이터(Terminator)의 결함. 그것을 교체하라(드라이브 시스템에는 단지 하나의 터미네이터만이 존재한다. 터미네이터는 보통 드라이브가 A:나 B:의 어느 쪽을 지명하든 간에 케이블로 연결된 디스크드라이브의 끝에 있는 드라이브상에 마운트(Mount) 된다).

●헤드의 미스얼라이먼트(Misalignment) 드라이브를 수선해야 한다.

●헤드의 불결함. 해당부분을 깨끗이 닦는다.

●헤드의 결함. 드라이브를 전문회사에 수선해야 한다.

●디스크드라이브 프레셔 패드(Pressure Pad)의 마모(싱글-사이드드라이브). 그것을 교체하거나 드라이브를 수리한다.

●프레셔 패드 암(Arm)(싱글 사이드드라이브)에 있는 스프링의 느슨함. 그것을 교체하거나 드라이브를 수리하러 보내라.

●인덱스 센서(Index Sensor)의 타버림. 드라이브를 수리하러 보내라.

●FDC보드 또는 디스크드라이브 로직보드의 결함. 가능하면 교체를 하고, 그렇지 않으면 드라이브를 수리하러 보내라.

●시스템 ROM의 결함. ROM을 교체하라. 이 문제는 흔하지 않다.

●CPU, 클럭 버스 컨트롤러 또는 다른 시스템보드 구성부품의 결함. 가능한 곳의 칩들을 고장수리하거나 교체한다. 이 문제 또한 흔치않다.

●시스템을 액세스할 때 2개의(또는 그 이상) 디스크드라이브에 동시에 비지 라이트(Busy Light)가 켜진다.

—디스크 드라이브 케이블의 플러그를 잡아 당긴다. 그것을 제거한 다음 다시 접속해 보고 케이블 코넥터상의 핀 1이 FDC 보드 및 디스크드라이브 로직보드의 핀 1에 적절히 맞게 접속되었는 지를 확인한다.

—디스크 드라이브가 부적당하게 어드레스되어 있다. 어드레싱 션트(Shunt)와 리어드레스(Readdress)를 정확하게 조사한다.

●디스크를 끼우거나 제거하는데 잘되지 않는다.

—헤드가 리드/라이트 위치로 부터 자유롭지 못하다. 시스템을 끈다음 켜본다. 스크래취 디스크(Scratch Disk)를 끼워 넣어 시도한다.

—스프링 또는 캐취가 절단되었다. 시스템을 끈다. 그다음 디스크드라이브를 열어 로직보드를 제거한 다음, 어떤 문제가 발생했는지를 조사하기 위해 스크래취 디스크(Scratch Disk)를 사용한다. 가능하다면 고장을 수리하고 그렇지 않으면 드라이브를 수리하러 보내라.

—드라이브 허브(Drive Hub)가 허브 링(Hub Ring)으로 부터 유출된 접착제에 의해 달라붙는다. 허브를 라이터 액(Lighter Fluid)으로 적신 헝겊으로 조심스럽게 닦아낸다.

—라이트-프로텍트(Write-Protect)나 다른 라벨(label)이 디스크 패스웨이(Pathway)에 다소 달라 붙는다. 그것을 제거하고 접착제를 조심스럽게 닦아낸다.

그밖에 알아야 할 사항

가능한 곳에서는 여러가지 일을 동시에 조사한다. 예를 들어 드라이브모터와 헤드의 청결상태, 그리고 헤드 캐리지 레일(Head Carriage Rail)의 청결상태를 거의 동시에 쉽게 조사할 수 있다. 어쨌든 정확한 판단에 따라 우선 주위에 있는 문제의 원인에 거의 가깝게 접근한다.

디스크드라이브내의 어떤 것도—윤활유를 쳐야 할 굳어있는 드라이브모터를 제외하고—힘을 가해서는 안된다. 여기서 힘을 가하는 것이란 매우 부드러운 종류의 것이야 한다.

모터 새프트(Motor Shaft)가 모터 하우징(Motor Housing)에 들어가는 장소와 새프트가 모터 하우징 위에 보이는 장소에 윤활유를 조금씩 떨어 뜨려야 한다. 어떤 다른 곳에 윤활유를 치는 것은 허용되지 않는다.

여분의 디스크드라이브를 사두면 동작중인 것이 깜박거릴 때 문제가 없게 된다. 두 세번의 보수 비용이면 새로운 드라이브로 교체될 수 있다. 그리고 반드시 같은 상표의 것을 구입할 필요는 없다. PC

터보 파스칼의 강력한 기능들

트리구조로 된 디렉토리와 자유롭게 이동, 삭제되는 윈도우는 터보 파스칼의 위력을 단적으로 보여준다.

볼랜드에서 제공되는 터보 파스칼은 현재 PC에서 최상의 프로그래밍 툴로써 인식되고 있다. 이것은 가장 저렴하고 가장 신속하며, 그리고 가장 콤팩트한 오브젝트 코드를 구현하고 있기 때문일 것이다. 또한 내장된 풀 스크린 에디터는 컴파일러가 발견한 에러를 사용자에게 알려주는 편리한 기능을 제공하고 있다.

터보 파스칼을 사용하여 사용자는 다른 기종의 시스템이나 오퍼레이팅 시스템에서도 액세스할 수 있으며 변수들은 특정 메모리 어드레스에 놓이게 되고 데이터는 바이트나 비트로 나누어 지며, 인터럽트는 서브루틴 호출로 수행된다. 또한 PC의 상태들을 액세스하는 능력도 프로그램에 부여할 수 있는데 이것은 대부분의 다른 언어에서는 쉽게 구현할 수 없는 것들이다.

터보 파스칼로 작성한 이러한 기능의 서브루틴들은 다른 종류의 파스칼 컴파일러로도 활용이 가능하며 C 또는 어셈블리어와 같은 다른 언어를 이용하는데도 활용이 가능하다.

이러한 루틴은 어떻게 작업을 수행하는지 즉, 적당한 장소에서 프로그램 내로 리스트를 복사하고 특정 방법으로 리스트를 호출하는 것에 대한 지식을 갖고 있지 않아도 사용이 가능하게끔 구성이 된다.

파라미터 전달

터보 파스칼 에디터에서 CTRL-K CTRL-R*은 다른 화일을 편집하고 있는 화일 속으로 삽입하는 것으로, 이것은 서브루틴을 불러내는데 편리한 방법이다. 즉 서브루틴은 필요에 따라 활용, 수정 및 삭제를 쉽게 할 수 있어야 하므로 터보 파스칼 에디터상에서 이같은 기능을 마련하게 된 것이다.

또한 $I 컴파일러 지시자**는 컴파일 진행 도중 다른 화일로부터 서브루틴을 읽어내는데 사용된다. 각 화일간에 중복되는 이름은 존재하지 않으며 전체적으로 배열된 이름들은 리스트의 시작부분에서 인식되기 시작한다.

터보가 생성한 컴파일 된 목적프로그램은 .COM화일인데 이 화일은 명령으로 그 화일명(.COM을 생략한)을 입력하면 자동적으로 불러들여 진다.

예를 들어, a : myprog. com이라는 화일을 다음과 같이 입력하면 불러낼 수 있다.

```
A>MYPROG
```

파라미터들을 코맨드라인으로부터 프로그램에 패스하는 기능이다. 즉 다음과 같은 입력

```
A>MYPROG HELLO
```

에 의해 사용자는 스트링 "HELLO"를 자신의 프로그램에 전달할 수 있게 된다. 프로시듀어 getparm(리스트 1 참조)이 바로 이런 역할을 보여주고 있다.

하나의 인수, parmtype(예를 들면 string[127])의 변수를 취하여 코맨드라인에서 읽혀지며 대기 파라미터가 존재하지 않으면 공백스트링을 리턴한다. 예를 들면 프로그램이 리스트 1과 다음 루틴으로 구성되어져 있을 때

```
Var  x : parmtype;
begin
   repeat
     getparm(x);
     writeln(x)
   until  x = ''
end.
```

*이 명령은 화일을 현재 커서가 있는 텍스트로 읽어들이는데 이용된다. 이 명령을 내릴 때에는 읽어들일 화일의 이름을 지정해야 되는데 화일이름은 규칙에 맞게 해야 한다. 확장자명이 생략되면 .PAS로 인식한다.

**프로그램의 작성을 손쉽게 해주는 이 컴파일러 지시자 I는 자주 사용되는 프로시듀어나 함수 등의 부프로그램을 별도로 작성하여 디스크에 기억시킨 후 주프로그램에서 필요로 할 때마다 원하는 위치에 삽입되어 동시에 컴파일을 시켜주는 기능을 갖는다. 코딩 서식은 {$I 화일이름} 으로 한다.

〔리스트 1〕 getparm루틴

```
{Turbo Pascal procedure to retrieve command line parameters }

 type parmtype = string[127];

 procedure  getparm(var  s:parmtype);

   { Returns first available parameter from DOS command }
   { line and removes it so next parameter will be      }
   { returned on next call. If no more parameters are   }
   { available, returns a null string.                  }

 var  parms: parmtype absolute CSEG:$80;
  begin
    s:= ' ';
    { parms[1] exists even when length is zero }
    while (length(parms) > 0) and (parms[1] = ' ') do
       delete(parms,1,1);
         while (length(parms) > 0) and (parms[1] <> ' ') do
           begin
             s:= s+parms[1]; delete(parms,1,1)
           end
  end;
```

이것을 다음 명령어로 불러낸다면

```
A>MYPROG ALPHA BETA GAMMA
```

다음과 같이 프린트한 다음 정지한다.

```
ALPHA
BETA
GAMMA
```

만일 주프로그램이 모든 파라미터들을 검색한 후에도 getparm의 호출상태를 계속 유지한다면 그때마다 빈 스트링을 얻게 된다.

각각의 파라미터는 앞이나 뒤에 공백없이 가변길이 스트링으로 리턴되기 때문에 가령 a : mydata.txt와 같이 변수가 화일명을 나타내는 경우를 대비해서 터보 파스칼은 assign이라는 화일조작용 프로시듀어를 내장하고 있다. 이 assign 프로시듀어는 지정한 화일명을 화일명변수의 값으로 할당하여 차후 이곳에서 지정한 화일명변수에 대한 모든 실제적인 조작은 지정된 화일이름을 대상으로 하게 된다.

또 문자열 처리 프로시듀어로 Val이라는 프로시듀어가 있는데 이는 지정 문자열을 수치로 변환시켜 준다. 단 문자열은 수치형식으로 표시된 문자열이어야 한다.

파라미터 리스트를 얻기 위해 getparm은 특별한 메모리배치로 할당된 변수에 의존한다. DOS가 프로그램을 로드할 때 프로그램의 최초 바이트를 코드세그먼트내의 옵셋 100H번지에 놓는다. 하위어드레스인 CSEG:0H에서 CSEG:FFH까지는 프로그램 세그먼트 프리픽스(PSP:Program Segment Prefix)라는 작업영역을 포함하며, 이 PSP내의 CSEG : 80H부터는 코맨드라인 파라미터들을 포함하는 128바이트이다. 그 최초 바이트는 나머지 127바이트들 중에 의미있는 데이터가 얼마나 많은 가를 나타내는 수로써 인식된다. 이것은 터보 파스칼이 어떻게 가변길이 스트링을 표현하고 있으며, 파라미터 리스트를 스트링으로써 CSEG : 80H의 첫번째 바이트(길이 지시자를 포함하는)에 어떻게 정의하는지를 알게 해준다.

이렇게 하여 스트링은 보통의 경우에 터보 파스칼에서 행해지는 스트링조작에 적합하게 되는데, 쉽게 말하면 스트링앞의 공백들과 스트링으로부터 읽혀지는 파라미터들을 삭제하여 스트링이 빌(공백) 때까지, 호출할 때마다 스트링이 점점 짧아지도록 하는 것이다.

리스트 1에서 주의할 것은 값이 길이가 0인 스트링일 지라도 parms〔1〕로 설정하고 있다는 점이며 프로시듀어 getparm의 내부에 있는 두개의 루프조건 테스트에서도 parms〔1〕으로 설정된 것을 볼 수 있다. 즉 터보 파스칼에서는 가변길이 스트링을 0으로 시작하여 차례로 번호가 지정되는 캐릭터들의 배열로서 표현한다. 그리고 바이트 0(예를 들어 parms〔0〕)에는 스트링의 길이를 나타내는 수의 값을 포함하고, 스트링 자체는 바이트 1에서 시작하게 된다. 그래서 string〔10〕으로 선언된 변수는 실제로 array〔0..10〕 of char이고 배열요소 0는 배열중에 얼마나 많은 데이터가 유효한가를 나타낸다.

Getparm은 MS-DOS나 PC-DOS의 어떤 버전에서도 수행될 수 있다. 만일 이것이 다른 컴파일러를 사용하는데 응용된다면 화일들을 열기전에 사용자의 프로그램속의 모든 파라미터들을 알맞게 수정시킬 필요가 있으며 또 어떤 형태의 프로그램들은 디스크화일이 오픈됐을 때 파라미터영역에 겹쳐질 때가 있다.

하지만 터보 파스칼에서는 이런 일이 전혀 없으므로 안심해도 된다.

클럭의 세트

보통 응용프로그램에서 날짜 및 시간을 프린트시 표제부나 또는 스크린디스플레이에 나타내기 위해 이들을 세팅해야 할 때가 있다. 베이직에서는 쉽게 세트가 되지만 터보 파스칼에서는 해당 프로시듀어를 사용자가 마련해야 한다. 리스트 2가 그것인데 함수 date과 time은 '01/01/89'와 '12:31:19'의 형태로 여덟개의 캐릭터스트링으로 받아들이고 다음의 명령

〔리스트 2〕 datetime루틴

```
{ Turbo Pascal routines to read and set date and time }

{ Each routine requires the following type definitions}
{ but does not require the other routines             }

type datetimetype = string[8];
    regtype       = record
                     ax,bx,cx,dx,bp,si,di,ds,es,flags : integer
                    end;

function date : datetimetype;
  { Returns current date in form '01/01/89'. }
var   reg : regtype;
      y,m,d,w: datetimetype;
      i:     integer;

begin
    reg.ax:=$2A00;
    intr($21,reg);
    str(reg.cx:4,y);
    delete(y,1,2);
    str(hi(reg.dx):2,m);
    str(lo(reg.dx):2,d);
    w := m + '/' + d + '/' + y;
    for i := 1 to length(w) do if w[i]=' ' then w[i]:= '0';
    date := w
end;

function time: datetimetype;
  { Returns current time in form '08:13:59'.}
var   reg:  regtype;
      h,m,s,w: datetimetype;
      i:    integer;

begin
  reg.ax := $2C00;
  intr($21, reg);
  str(hi(reg.cx):2,h);
  str(lo(reg.cx):2,m);
  str(hi(reg.dx):2,s);
  w := h + ':' + m + ':' + s;
  for i:= 1 to length(w) do if w[i] = ' ' then w[i] := '0';
  time := w
 end;

procedure  setdate(x:datetimetype);
  { Set date, Accepts string in format '01/02/89'. }
var   reg:      regtype;
      rh,rl,c1,c2,c3: integer;
begin
   reg.ax := $2B00;
   val(x[1] + x[2], rh, c1);  { month goes in Ln }
   val(x[4] + x[5], rl, c2);  { day goes in DL   }
   reg.dx := rh * 256 + rl;
   val(x[7] + x[8], rl, c3);  { year goes in CX  }
   reg.cx := rl + 1900;
   if rl < 80 then reg.cx := reg.cx + 100;  { 21st century }
   c1 := c1 + c2 + c3;        { return codes from VAL }
   if c1 = 0 then intr($21,reg);
   if c1 + lo(reg.ax) <> 0 then
      begin
        writeln;
        writeln('Error--Invalid date,''',x,'''');
        halt
      end
 end;

 procedure settime(x:datetimetype);
   { Set time, Accepts string in format '09:13:59'. }
 var   reg:  regtype;
       rh,rl,c1,c2,c3: integer;
 begin
    reg.ax := $2D00;
    val(x[1] + x[2], rh, c1);  { Hours go in CH }
    val(x[4] + x[5], rl, c2);  { Minutes go in CL }
    reg.cx := rh * 256 + rl;
    val(x[7]+x[8],rh,c3);      { Seconds go in DH }
    reg.dx := rh * 256;
    c1 := c1 + c2 + c3;        { Return codes from VAL }
    if c1 = 0 then intr($21,reg);
    if c1 + lo(reg.ax) <> 0 then
      begin
        writeln;
        writeln('Error--Invalid time, ''',x,'''');
        halt
      end
end;
```

```
writeln(DATE,' ',TIME)
```

이 실행되면 컴퓨터는 날짜 및 시간을 디스플레이 한다. 그리고 프로시듀어 setdate와 settime은 이와는 반대로 수행되는데, 이들은 date와 time에서 사용된 것과 같은 포맷인 스트링을 받아들여서 클럭을 세트한다. 이들의 인수는 상수 또는 변수가 될 수 있는데

```
setdate('09/20/89')
```

그리고

```
var  x : string[8];
...
writeln('What is the date ?');
readln(x);
setdate(x)
```

은 모두 유효하다. Setdate와 settime은 무효한 인수를 받으면 프로그램을 중지시켜버리지만 원한다면 변경 시킬 수 있으며 DOS수준에서는 무효한 인수가 아무런 해를 끼치지 않으며 클럭에도 영향을 주지 않는다. 또 각 루틴들의 시작부분에서 타입정의를 선언하고 있지만 서로를 호출하는 경우는 없으므로 리스트 2가 프로그램에 내장된 후 사용되지 않는 다른 루틴들은 삭제해도 상관없다.

Date, time, setdate, settime은 DOS가 제공하는 인터럽트 서비스루틴을 액세스함으로써 작업을 수행한다. 여기서 사용되는 모든 DOS서비스는 인터럽트 21H를 통해 액세스되고 AH 레지스터의 내용들은 어떤 서비스가 수행되는 지를 결정하지만 다른 레지스터들은 인터럽트 서비스루틴으로부터 정보를 전달해 준다.

터보 파스칼은 어떤 인터럽트를 수행하는 내장프로시듀어 intr을 갖고 있다. 이것은 2개의 파라미터 즉 인터럽트 인식번호와 레지스터로 로드되는 값들의 레코드를 취한다. 리스트의 처음부분에서 정의된 타입 regtype는 이러한 레코드의 구조를 설정하고 레지스터명을 그의 구성성분으로 관련시키는데 intr은 레코드를 레지스터로 복사하고, 인터럽트를 수행하며, 레지스터를 레코드로 다시 복사하여 호출프로그램이 레지스터를 복원할 수 있도록 한다.

각 레지스터(예로 AX)는 2바이트 정수형으로서 취급된다는 것에 유의하라. 하프레지스터(예로 AH, AL)를 조사할 필요가 있을 때 LO와 HI는 각각 상·하위값을 제공한다. Date와 time은 이러한 기법을 폭넓게 사용한다.

레지스터의 로우와 하이 바이트내로 값들을 로드하는 한 방법은 하이값에 256(100H)를 곱한 값을 로우값에 더한 다음 그 결과를 전체 정보로 할당하는 것이며, 프로시듀어 setdate와 settime은 여러 곳에서 이 일을 수행한다.

인터럽트 서비스는 DOS 테크니칼 레퍼런스 매뉴얼에 상세히 설명되어 있다. 말하자면 함수 date는 요구되는 서비스가 어떤 종류인가를 인식하고 인터럽트를 수행한 후 레지스터 값들을 수정하고 레지스터내에 있는 번호들을 내장 Str 프로시듀어를 이용하여 스트링으로 변환한다. Time도 이와 매우 유사하다.

Setdate와 settime은 이와 반대로 작용하는데 즉, 이들은 스트링을 val을 이용하여 번호로 변환하고 레지스터에 적당한 값들을 로드하고 인터럽트를 수행한다. 최종적으로는 에러를 발견한 것이 val인지 또는 인터럽트 그 자체인지를 체크한다. 최종 단계까지 에러체크를 미루는 것이 좋은데 그 이유는 무효한 인수가 인터럽트 서비스루틴에 의해 체크되며 클럭에 아무런 영향을 끼치지 않기 때문이다.

트리구조의 디렉토리

DOS 2.0과 그 이상 버전의 큰 매력은 유저에게 서브디렉토리를 만들고 화일을 트리와 비슷한 구조로 체계화 할 수 있는 능력을 제공한다는 점이다. 화일은 다음의 예와 같이 이름에 의해서가 아니라 패스에 의해 연관된다.

```
A:\alpha\gamma\myprog.pas
```

이와 같이 드라이브 문자와 화일의 실제 배치에 따라 써내려간 디렉토리명의 체인으로 구성된다. 만일 '\'(드라이브 문자 다음)으로 시작된다면 패스는 디스크의 루트 디렉토리에서 출발하며, 그렇지 않으면 커런트 디폴트 디렉토리에서 시작되는데 여기서 화일은 이름(패스 없이)에 의해 연관된다는 것을 생각할 수 있다.

2.0 이상의 DOS버전과 호환성을 갖기 위해 터보 파스칼은 단지 화일명만에 의해 화일을 액세스함으로써 커런트 디렉토리내에 있는 화일들에 도달할 수 있다. 터보 환경은 한 디렉토리에서 다른 디렉토리로 이동하는 명령을 제공하지만 프로그램에서 그와 같은 이동에 대한 방법을 제공하지는 않는다.

리스트 3은 이와 같은 문제를 전 범위의 디렉토리 조정 루틴을 제공함으로써 해결하고 있다. 여기서 rtype는 날짜 및 시간의 루틴에서 regtype과 같은 것이지만 이름이 다른 것은 두개의 루틴이 동시에 동일 프로그램에 삽입될 때 중복을 막기 위함이다.

프로시듀어 ××diskerr과 ××patherr는 다른 루틴들의 에러메시지를 출력하는 곳이다. 나머지 프로시듀어들은 인터럽트 21H를 통해 시스템을 처리하고 있다. 대부분의 경우에 서비스들은 단지 DOS버전 2.0과 그 이상에 의해 제공되지만 currentdrive와 chdrive는 예외인데 이들은 DOS의 모든 버전에서 작업을 할 수 있다. Currentdrive는 뒤에 콜론(:)이 따르는 커런트 드라이브문자를 리턴하는 기능이다. 예를 들어 현재 A:가 선택되었고 명령

```
writeln(currentdrive)
```

이 실행되면 컴퓨터는 A:를 프린트할 것이다. Chdrive는 이와 반대로 수행하는데 이것은 대문자나 소문자이던 콜론이 있든지 없든지간에 같은 형식의 인수를 받아 들인다. 예로

```
chdrive('A:')
```

〔리스트 3〕 treedir루틴

```
{ Turbo Pascal routines for tree-structured directories }
{ Requires MS-DOS or PC-DOS 2.0 or higher, except as noted. }

{ All the routines require these type definitions.          }
{ However, except as noted, they do not require each other. }

type pathtype = string[63];
     drivetype = string[2];
     rtype     = record
                 ax,bx,cx,dx,bp,si,di,ds,es,flags: integer
                end;

procedure xxdiskerr(x:drivetype);
  begin
     writeln('Error -- Invalid disk drive, ''',x,'''');
     halt
  end;

 procedure xxpatherr(x:pathtype);
 begin
   writeln('Error -- Invalid path, ''',x,'''');
   halt
 end;

function currentdrive: drivetype;
  {Returns designator for current default drive, e.g., 'A:'. }
  {Works under DOS version 1. }
var
  w:  drivetype;
  reg: rtype;
begin
   reg.ax := $1900;
   intr($21,reg);
   w:= 'A:';
   w[1]:= chr(ord(w[1]) + lo(reg.ax));
   currentdrive := w
end;

procedure chdrive(x:drivetype);
   { Chooses a new default drive. }
   { parameter can have the form 'A:', 'A', 'a:', 'a'.      }
   { Works under DOS version 1. Requires XXDISKERR, above. }
var
   reg: rtype;
begin
   reg.ax := $0E00;
   reg.dx := ord(upcase(x[1])) - ord('A');
   intr($21,reg);
   if (reg.dx < 0) or (lo(reg.ax) < lo(reg.dx)) then xxdiskerr(x);
end;

function  diskspace(x:drivetype): real;
   { Returns number of byes available on specification disk.  }
   { Parameter as for CHDRIVE. Requires XXDISKERR, above.     }
var
   reg: rtype;
begin
   reg.ax:= $3600;
   reg.dx := 1 + ord(upcase(x[1])) - ord('A');
   intr($21,reg);
   if reg.ax = $FFFF then
      xxdiskerr(x)
   else
      diskspace := (256.0*hi(reg.dx)+lo(reg.dx))*reg.ax*reg.cx
end;

function  currentdir(x:drivetype): pathtype;
  { Returns full path to active directory on specified drive, }
  { including backslash at beginning, not including drive     }
  { designator. Parameter as for CHDRIVE.                     }
var
  w    : pathtype;
  reg  : rtype;
  i    : integer;
begin
   { Get current path }
   reg.ax := $4700;
   reg.dx := 1 + ord(upcase(x[1])) - ord('A');
   reg.ds := seg(w[1]);
   reg.si := ofs(w[1]);
   intr($21,reg);
   if (reg.flags and 1) > 0 then xxdiskerr(x);

   { Turn it into a Turbo string }
   i := 1;
   while w[i] <> chr(0) do i:= i + 1;
   w[0] := chr(i-1);
   for i:= 1 to length(w) do w[i] := upcase(w[i]);

     currentdir := '\' + w
end;

procedure xxdir(x:pathtype; k:integer);
   { Execute CHDIR, MKDIR, and RMDIR requests.    }
   { Requires XXPATHERR and CURRENTDRIVE, above. }
var
   w: pathtype;
   reg: rtype;

begin
  w:= x + chr(0);
  if w[2] <> ':' then     { add drive designator }
```

```
      w:= currentdrive + w;
   reg.ax := k;
   reg.ds:=  seg(w[1]);
   reg.dx := ofs(w[1]);
   intr($21, reg);
   if (reg.flags and 1) > 0 then xxpatherr(x)
 end;


 procedure   chdir(x:pathtype);
   { Equivalent to CHDIR command in DOS.              }
   { Requires XXDIR, XXPATHERR, and CURRENTDRIVE, above. }
   { Caution! Do not leave a directory  }
   { if you have files in it open.      }
 begin
    xxdir(x,$3B00)
 end;

  procedure rmdir(x:pathtype);
    { Equivalent to RMDIR command in DOS.               }
    { Requires XXDIR, XXPATHERR, and CURRENTDRIVE, above. }
  begin
    xxdir(x,$3A00)
  end;

 procedure mkdir(x:pathtype);
    { Equivalent to MKDIR command in DOS.               }
    { Requires XXDIR, XXPATHERR, and CURRENTDRIVE, above. }
 begin
    xxdir(x,$3900)
 end;

 procedure  rename(x,y:pathtype);
    { Renames a file; unlike the DOS RENAME command,  }
    { both parameter of this command are full paths.  }
    { The paths need not be the same, allowing a file }
    { to be moved from one directory to another.      }
    { First parameter can specify a drive; any drive  }
    { letter on the second parameter is ignored.      }
 var
   wx,wy: pathtype;
   reg  : rtype;
 begin
   wx := x + chr(0);
   wy := y + chr(0);
   if wx[2] <> ':' then wx := currentdrive + wx;
   reg.ax := $5600;
   reg.ds := seg(wx[1]);
   reg.dx := ofs(wx[1]);
   reg.es := seg(wy[1]);
   reg.di := ofs(wy[1]);
   intr($21,reg);
   if (reg.flags and 1) <> 0 then
     begin
       writeln('Error -- Invalid rename request');
       writeln('      -- From: ''',x,'''');
       writeln('      -- To:   ''',y,'''');
       halt
     end
 end;
```

는 디폴트드라이브를 A:로 바꾼다.

Diskspace는 해당 디스켓에서 사용가능한 바이트 용량을 구하는 루틴이다. 이것은 chdrive와 같은 인수를 취하는데 다음과 같은 경우이다.

```
writeln(diskspace('A:'):10:0,'bytes free on drive A:')
```

Writeln문의 ':10:0'은 diskspace의 값이 실수형태이기 때문에 필요한 것이다. Currentdir은 해당 디스크의 커런트디렉토리로 패스를 리턴하는 것만 제외하고는 currentdrive와 비슷하다. 이것 역시 드라이브문자를 제외한 백슬래쉬로 시작하는 전체의 패스이다. 예로 currentdir('B:')의 값은 '\myprog\wrkspace'으로 된다. 당연히 이것은 화일명이 아니라 디렉토리명으로 끝나게 된다.

3개의 루틴 chdir, rmdir, mkdir은 같은 형태의 DOS명령들과 유사하다. 즉 chdir은 커런트디렉토리로 바꾸고 rmdir은 디렉토리(어떠한 화일도 포함하지 않아야 함)를 삭제하며, 그리고 mkdir은 디렉토리를 만드는 것이다. 이들은 각각 하나의 파라미터로 63개까지의 캐릭터 스트링을 취하는데, DOS명령 수준에서의 파라미터와 같은 방법으로 해석된다. 또한 이들 3개의 운영을 위해 필요한 DOS호출은 AH레지스터의 값에 일치시켜 3개의 모든 루틴들의 파라미터를 포맷하고 인터럽트를 수행하기 위해 프로시듀어 ××dir을 이용한다.

Rename 프로시듀어는 DOS에 있는 명령과 유사하지만 이보다 더 강력하다. DOS에서 rename은 단지 화일의 이름만을 바꾸지만 이 rename은 또한 화일을 한 디렉토리에서 다른 디렉토리로 옮길 수 있다. 즉

```
rename('A:\alpha\myprog.pas','\beta\mypro.pas')
```

은 myprog.pas을 디렉토리 alpha에서 디렉토리 beta로 옮긴

다. 그러나 아직도 이것은 한 디스크에서 다른 디스크로 화일을 옮기지 못하지만, 만일 드라이브 문자가 두번째 파라미터라면 이 문자는 무시된다.

편리한 윈도우조작

터보 파스칼을 이용하여 유저들은 출력을 스크린에 편집하여 나타내거나 부분화면 및 한정된 스크린의 특정영역에 스크롤링과 같은 방법으로 윈도우를 정의해서 쓸 수 있다.

리스트 4에서의 루틴들은 터보 파스칼에서 쉽게 제거가 가능한 윈도우를 만들어 준다. 윈도우가 생길 때마다 스크린 전체의 내용들이 커서위치와 윈도우 크기에 따라 지정된다. (화면 1 참조). 물론 윈도우가 클로우즈 됐을 때는 이전의 내용들이 복원돼야 함은 당연하다. 윈도우의 중첩은 상수 maxwin에 세트되는 갯수만큼 가능하고 단지 제약점은 스크린의 각 영상을 저장하는데 4kB 이상의 메모리를 필요로 한다는 측면에서의 메모리문제밖에 없다. 리스트 5에서는 전체의 루틴을 테스트하여 처리하고 있다.

〔리스트 4〕 window루틴

```
{ Turbo Pascal removable window system }

{ Requirements: IBM PC or close compatible.   }
{ Screen must be in text mode, on page 1,     }
{ either mono or color card.                  }

{ Call INITWIN before calling MKWIN or RMWIN. }

const maxwin = 5; {maximum number of windows open at once }

type imagetype = array[1..4096] of char;
     windimtype = record
                   x1,y1,x2,y2 : integer
                  end;
var
  win : { Global variable package }
     record
        dim : windimtype ;        {Current window dimensions }
        depth : integer;
        stack : array[1..maxwin] of
                 record
                   image : imagetype; { Saved screen image       }
                   dim   : windimtype;{ Saved window dimensions }
                   x, y  : integer    { Saved cursor position   }
                 end
       end;

crtmode     : byte      absolute $0040:$0049;
crtwidth    : byte      absolute $0040:$004A;
monobuffer : imagetype absolute $B000:$0000;
colorbuffer: imagetype absolute $B800:$0000;

procedure initwin;
begin
   with win.dim  do
     begin  x1:=1;  y1:= 1; x2:= crtwidth; y2:= 25 end;
     win.depth := 0
  end;

procedure boxwin(x1,y1,x2,y2: integer);
   { Draws a box, fills it with blanks, and makes it the current }
   { window. Dimensions given are for the box; actual window is  }
   { one unit smaller in each direction.                          }
   { This routine can be used separately from the rest of the    }
   { removable window package.                                     }
var
  x,y: integer;
begin
   window(1,1,80,25);
   { Top }
   gotoxy(x1,y1);
   write(chr(213));
   for x:= x1+1 to x2-1 do write(chr(205));
   write(chr(184));

   { Sides }
   for y:= y1+1 to y2-1 do
     begin
        gotoxy(x1,y);
        write(chr(179), ' ':x2-x1-1, chr(179))
     end;

   { Bottom }
   gotoxy(x1,y2);
   write(chr(212));
   for x:= x1+1 to x2-1  do write(chr(205));
   write(chr(190));

   { Make it the current window }
   window(x1+1,y1+1,x2-1,y2-1);
   gotoxy(1,1)
 end;


procedure   mkwin(x1,y1,x2,y2: integer);
   { Create a removable window }
```

```
begin
  { Increment stack pointer }
  with win do depth := depth +1;
  if win.depth > maxwin then
    begin
      writeln(^G,' Windows nested too deep ');
      halt
    end;

   { Save contents of screen }
   if crtmode = 7 then
      win.stack[win.depth].image := monobuffer
   else
      win.stack[win.depth].image := colorbuffer;

   win.stack[win.depth].dim := win.dim;
   win.stack[win.depth].x   := wherex;
   win.stack[win.depth].y   := wherey;

   { Create the window }
   boxwin(x1,y1,x2,y2);
   win.dim.x1 := x1 + 1;
   win.dim.y1 := y1 + 1; {Allow for margins }
   win.dim.x2 := x2-1;
   win.dim.y2 := y2-1;
end;

procedure rmwin;
  { Remove the most recently created removable window }
  { Restore screen contents, window dimensions, and   }
  { position of cursor.                               }
begin
  if crtmode = 7 then
     monobuffer := win.stack[win.depth].image
  else
     colorbuffer := win.stack[win.depth].image;
  with win do
    begin
      dim := stack[depth].dim;
      window(dim.x1,dim.y1,dim.x2,dim.y2);
      gotoxy(stack[depth].x, stack[depth].y);
      depth := depth - 1
    end
end;
```

[리스트 5] windemo루틴

```
{ Demonstration program for removable window package }

{ --- Insert code from Listing 4 here --- }

var
  i : integer;

begin
  initwin;
  writeln('Now and every time the action stop,');
  writeln('press ENTER to continue...');
  readln;
  clrscr;
  for i:= 1 to 25 do writeln(' This is the original screen.');

  mkwin(3,3,50,18);
  for i:= 1 to 15 do writeln('This is the first window...');
  readln;

  mkwin(10,5,70,20);
  for i:= 1 to 15 do writeln('Second window...');
  readln;

  mkwin(15,15,45,23);
  writeln('Third window...');
  readln;

  mkwin(55,10,79,25);
  writeln('Fourth window...');
  readln;

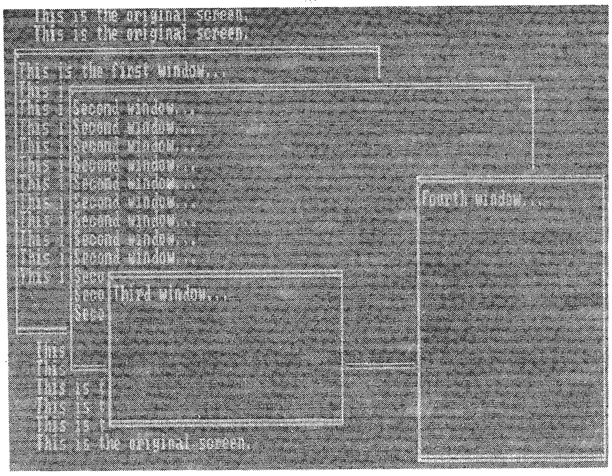
 rmwin;  { Removable fourth window }
 readln;

 rmwin;  { Remove third window }
 writeln;
 writeln('We are back in the second window...');
 readln;

 rmwin;  { Remove second window }
 writeln;
 writeln('This is the first window again !');
 readln;

 rmwin;  { Remove first window }
 readln
end.
```

〔화면 1〕 windemo.pas의 실행 예



스크린 내용들은 비디오메모리의 내용을 세이브함으로써 저장할 수 있고 이런 루틴은 IBM PC에서와 똑같은 메모리 어드레스를 사용하는 컴퓨터에서만 작업을 하게 된다.

프로시듀어 initwin은 전체 변수를 초기화하기 위해서 다른 어느 프로시듀어보다도 먼저 호출되어야 한다. 프로시듀어 mkwin은 윈도우를 생성하고 4개의 파라미터들을 취한다. 다음 문장

```
mkwin(5,2,30,15)
```

은 5컬럼, 2라인(왼쪽 상단을 기준)에서 30컬럼, 15라인까지 확장된 윈도우를 만들어 낸다. 실제의 작업영역은 각 방향으로 2개의 유니트가 더 작은 것인데 이는 2개의 컬럼과 라인의 경계로써 사용되기 때문이다. 커서는 윈도우의 제일 위의 왼쪽에서 출발한다.

Rmwin은 윈도우를 제거하는데 사용된다. 이것은 어떠한 파라미터도 취하지 않으며 내부적으로 mkwin은 경계를 긋고 제한 스크롤링 영역(터보 파스칼이 윈도우를 호출하는 것)을 정의하기 위해 프로시듀어 boxwin을 사용한다.

Boxwin은 독립적으로 사용될 수 있지만 boxwin과 내장 프로시듀어 window는 mkwin 및 rmwin을 사용하는 프로그램에서는 호출되지 않는데 왜냐하면 rmwin은 mkwin에 의해 만들지 않은 윈도우는 존재하지 않는다고 가정하기 때문이다.

윈도우 루틴은 그래픽모드가 아니라 텍스트모드에서만 수행이 되며 단지 디스플레이 페이지 0상에서만 작업을 수행한다. 여기서 윈도우 커서 위치와 크기 등은 저장하는 반면 현재 선택된 비디오 농도, 텍스트 칼라, 또는 디스플레이 폭 등은 저장하지 못한다는 점에 주의하기 바란다.

이제까지 설명한 루틴들은 터보 파스칼을 이용하여 무엇을 수행할 수 있는 가에 대한 예에 불과하지만, 일반적으로 터보 파스칼은 PC에서 할 수 있는 모든 것을 가능하게 해주는 언어로 부상하고 있다. PC

월간 퍼스널컴퓨터의 정기구독자에게는 아래와 같은 특전이 마련되어 있읍니다.

—정기구독자는 본사가 보유한 각종 정보 및 자료를 무료로 배부받으실 수 있읍니다.
—본지에 게재되는 각종 프로그램을 디스켓에 담아 무료로 보내드립니다.
—본지 및 본지의 자매지(월간 전자과학) 주최로 개최되는 세미나 및 각종 행사에 무료로 참석하실 수 있읍니다.
—본지에서 행하는 각종 부대사업에 할인혜택이 주어집니다.
—구독기간중 책 가격이 인상되더라도 종전가격으로 구독이 가능합니다.

●구독신청은

—시중은행에서 지로용지를 이용하시거나 우체국에서 대체구좌 또는 소액환으로 송금하시면 됩니다.
은행 지로번호 : 3000667
우체국 대체구좌번호 : 010256—31—0526772

●정기구독료는

—1 년 구독시 30,000원
—6개월 구독시 15,000원

●구독신청시 주의사항은

-구독기간과 주소(우편번호), 전화번호와 받으실 분의 성명을 정확히 기입하시기 바랍니다.
—배달사고나 기타 변동사항은 본지 영업부로 즉시 연락하시기 바랍니다. 신고전의 분실사고는 책임지지 않습니다.

●정기구독 문의처는

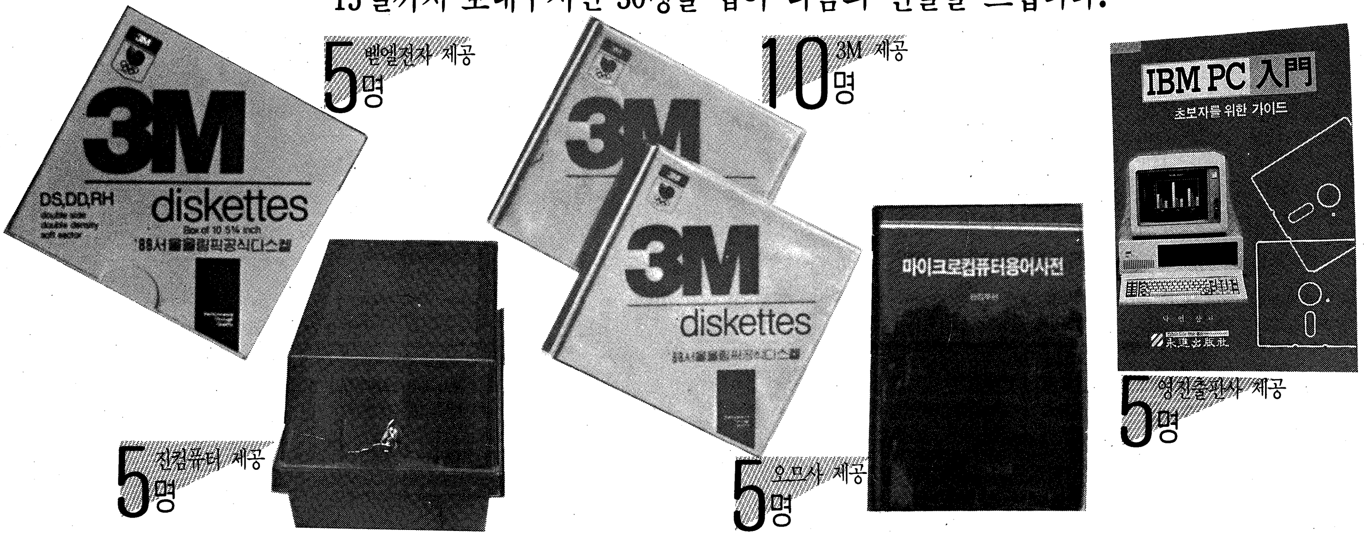
—서울시 용산구 갈월동 17 ㊥ 140-150
월간 퍼스널컴퓨터 영업부
전 화 : (02)754—1044(대표)
FAX : 757—1049

■ 전국총판 ■

●서울●	종로, 중구	명 문 사	774—6314
	성동, 동대문	홍익도서	446—3926
	성북, 도봉	진 원 사	981—4884
	강남, 강동	광 일 사	415—7330
	관악, 동작	관 악 사	889—4504
	서대문, 은평	형 제 사	304—9462
	영등포, 구로	영 도 사	613—5381
		진명서점	272—5871
		관음서적	266—7792
●부산●		종합기술서적	462—2431
		영광도서전시관	805—9500
		북부총판	805—9677
		남부총판	756—2727
●대구●		한일서적	23—7101
		학 원 사	22—0336
●인천●		동양서점	73—3384
●광주●		삼복서점	22—0258
●경기●	부천	대경서점	62—3672
	안양, 안산	사랑글방	57—1916
	수원,	동아서점	5—6436
	성남	영광서점	2—6532
●강원●	원주	원주종로서적	42—2943
	삼척	영 동 사	72—6060
	강릉	삼 일 사	2—2314
	춘천	대진서점	54—5148
●충북●	충주	문 학 사	2—2814
	청주	화 문 당	55—0503
●충남●	천안	동방서적공사	63—3314
	대전	동양서점	22—1538
●경북●	포항	경북서점	2—2717
	구미	중앙서점	52—7915
●경남●	진주	동아서점	42—3768
	진해	학애서점	2—2523
	울산	우정서점	43—6003
	마산	기술서적	56—1417
●전북●	전주	홍지서점	6—5311
	군산	동원서점	42—3776
●전남●	목포	문 화 당	5—3086
	여수	대양서점	62—2111
●제주●		광양서적	53—5077

애독자 선물대잔치

월간 퍼스널컴퓨터는 독자 여러분의 의견을 소중히 하여 책의 편집에 반영하고자 합니다.
애독자 설문지를 사용하여 질의내용에 성의껏 답하시어
15일까지 보내주시면 30명을 뽑아 다음의 선물을 드립니다.



● 당첨자 명단 ●

● 디스켓 보관함 ●

김종훈 서울 영등포구 여의도동 서울아파트 2동 805호
박 찬 광주직할시 동구 용산동 216 16/1
이태원 서울 구로구 오류2동 137-14 15/2
현동주 강원도 원주시 봉산1동 1088-7 (원용인씨댁)
홍상준 충남 온양시 온천1동 242-5

● 디스켓 박스(10장들이) ●

김남하 서울 강동구 명일동 339--26 8/3
김미현 서울 구로구 구로동 142-58 10/3
김진호 부산시 사하구 괴정1동 495-14 7/5
박진우 서울 동작구 흑석2동 25구획 1호 청명빌라 102호
최윤석 서울 은평구 역촌2동 81-61

● 컴퓨터 용어사전 ●

김성춘 서울 중구 예장동 8 리라공고 정보기술과
배정곤 서울 강동구 성내동 미주아파트 3동 307호
이경희 전남 여천시 학동 54 주공아파트 3동 407호
박정근 경기도 수원시 장안구 지동 127-7
이정만 경기도 과천시 주공아파트 641동 104호

● 디스켓 자켓(4장들이) ●

노진호 서울 은평구 불광동 16-88
김석철 서울 은평구 갈현동 518-14
윤은식 서울 성북구 삼선동 1가 83-2
이백진 경기도 광명시 광명6동 374-142 동방연립 12동 101호
이순규 강원도 원성군 소초면 장양리 176-3
임대규 부산시 해운대구 반여1동 1477-1 왕자아파트 14동 306호
전계진 강원도 원주시 우산동 상지실전 전자계산과
정희태 충남 대전시 동구 성남1동 177-210 3/5
황광신 충북 제천시 청전동 137-11 3/1
이우관 서울 성북구 정릉1동 123 스카이맨션 1동 102호

● 컴퓨터 다이어리 ●

김영태 충남 부여군 세도면 청포리 3구 93
윤원제 서울 송파구 신천동 장미아파트 8동 802호
이성철 경남 진해시 경화동 1149-33
이용복 광주시 월산5동 615-7 7/1
임삼열 서울 동작구 노량진1동 214-142 6/6

서울 거주자는 1월31일까지 신분증을 지참하시고 직접 본사로 나오시면 해당 상품을 드리며, 지방에 계신 분은 우송해 드리겠읍니다.
기간은 1월31일까지이며, 이후는 무효로 처리합니다.

※ 이곳에 풀칠을 하여 뒷면 수신란이 나오게 접어서 우체통에 넣어 주세요.

애독자 설문지

책의 편집에 반영하고자 하오니 성의껏 적어서 보내주시기 바랍니다.

작성자	이름		나이		직장		성별	
	주소	☎ (　　–　　)						

사용기종	8비트		16비트		32비트	
	주변기기	□ 모니터(컬러, 단색)　□ 드라이브(HDD, FDD)　□ 프린터(9핀, 24핀, 레이저)　□ 인터페이스카드(　　　)				
	사용용도	□ 취미　□ 업무　□ 교육　□ 학습　□ 기타			사용경력	
	사용언어	□ 베이직　□ C　□ 파스칼　□ 기계어　□ 기타				
	사용분야		사용 소프트웨어			

책 구입방법	구입방법	□ 정기구독　□ 서점구입(　　　　서점)　□ 증정　□ 기타		
	구입동기	□ 정기구독　□ 필요한 내용이 있어서　□ 주위에서 권해서　□ 기타		
	본지외 구독잡지	□ 없다　□ 있다(잡지명 :　　　　)	원하는 발행일자	
	애로사항	□ 가격이 비싸다　□ 종이 질이 나쁘다　□ 인쇄상태가 나쁘다　□ 기타		
	□ 본지 구독을 추천하고 싶은 사람이 있다면(이름과 주소) ☎ (　　–　　)			

이번달에 좋았던 기사(2가지)	제목		제목	
	□ 이유		□ 이유	

이번달에 싫었던 기사(2가지)	제목		제목	
	□ 이유		□ 이유	

보내는 사람

□□□-□□□

우 표

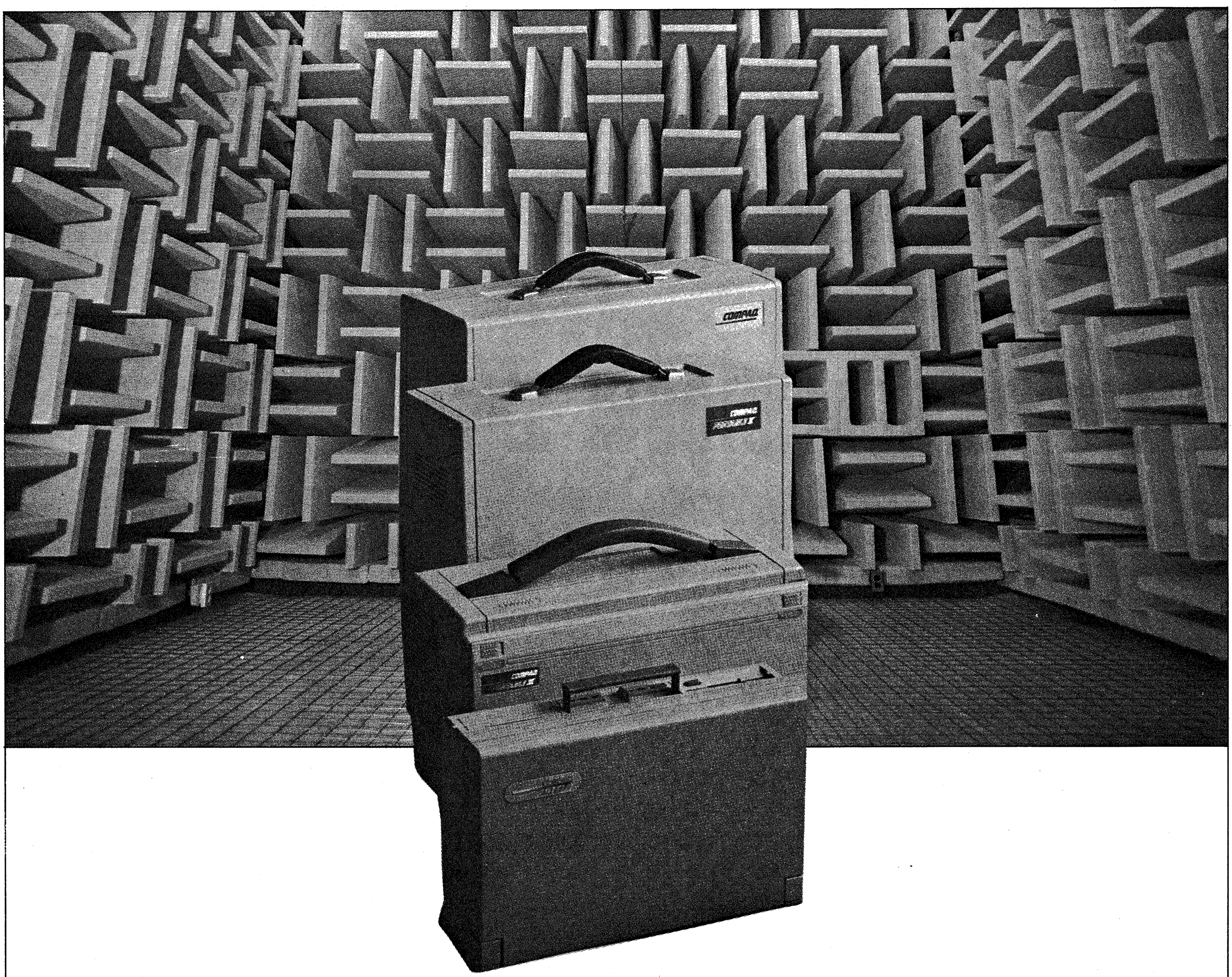
월간 퍼스널컴퓨터 하이테크

서울특별시 용산구 갈월동 17

퍼스널컴퓨터 편집부앞

140-150

애독자페이지	관심있는 분야			게재를 원하는 기사		
	☐ 컴퓨터에 관한 알고 싶은 질문					
	☐ 본지에 바라는 의견					
정기구독신청서	구독신청방법	☐ 은행지로	☐ 우체국 대체구좌	☐ 소액환	☐ 기타	
	구독기간	년 월호부터	년 월호까지	구독신청일자		
	☐ 받으실 분의 이름과 주소(우편번호) ☎ (—)					



□ 특집 □

IBM PC ROM BIOS의 분석

- IBM PC의 BIOS
- 중요한 BIOS들
- BIOS 사용상의 유의점
- MS-DOS의 특징

BIOS는 주변 LSI의 제어프로그램 등을 모듈화한 것으로
하드웨어를 직접 제어하는 프로그램의 모임이다.
따라서 BIOS의 내용과 그 사용방법 등을 알아두면
직접 I/O부분을 불러낼 수 있어 시스템을 이해하는데 도움이 된다.

IBM-PC ROM BIOS의 분석

BIOS는 주변LSI의 제어프로그램 등을 모듈화한 것이며, 하드웨어를 직접 제어하는 프로그램의 모임이다.

일반적으로 BIOS의 펑션 콜은 OS가 하므로 유저는 BIOS에 걱정할 필요는 없지만 MS-DOS의 기능이 그렇게 높지 않기 때문에 직접 프로그램에서 BIOS의 펑션콜을 하도록 하는 사용법이 증가하고 있다.

따라서 MS-DOS로 프로그램을 작성하도록 하는 환경에서는 BIOS의 내용을 알아 두면 편리하다.

IBM PC의 BIOS

IBM PC용으로 시판되는 대부분의 소프트웨어에서는 I/O디바이스의 핸들링이나 고속성이 요구되는 사양의 해결로서 BIOS펑션 콜을 간단히 하고 있는 예가 증가하고 있다. 통상 OS의 펑션 콜을 사용하지 않고 BIOS를 부르도록 하는 행위는 그 기종에 대해서 충분히 알고 있지 않으면 안되며 기종마다 BIOS가 다르기 때문에 사용시 주의를 요한다.

그러나 IBM PC시리즈의 소프트웨어 아키텍처를 보면 버전업이 이루어졌다고 하여도 OS의 하위에 해당하는 BIOS에 대해서 상위호환을 굳게 지키고 있기 때문에 장기간에 걸쳐서 안정된 호환성이 보증되고 있다.

특히 그래픽관계가 DOS에 의해서 소프트되어 있지 않기 때문에 응용 서포트웨어에서는 BIOS콜에 의지하지 않을 수 없다. MS-DOS는 그 이름대로 디스켓의 호환성밖에 보증하지 않는다고 할 수 있고, 일반 I/O콘트롤러의 핸들링마저 역부족이므로 BIOS야말로 시스템자체를 핸들링하기 가장 좋은 일이라고 할 수 있다.

즉, BIOS는 세심하게 배려하여 대단히 편리한 것이며, BIOS를 콜하면 여러가지의 기능을 포함한 프로그램을 어셈블러로 쉽게 쓸 수 있다. 최근은 C와 같은 고급언어의 라이브러리에 BIOS 기능콜이 표준 서포트되도록 있으므로 어셈블리어에 정통하여 있지 않아도 즐겁게 프로그래밍할 수 있다.

●BIOS-ROM의 내용

IBM PC는 주지하는 바와 같이 오픈 아키텍처라 하여 그 기술적 내용을 모두 공표하고 있다. IBM의 전략이라고 하는 사람도 있지만 IBM PC시리즈의 하드웨어 테그니컬 레퍼런스 매뉴얼에는 BIOS의 소스 리스트가 완전한 형으로 게재되어 있다.

BIOS-ROM의 기본적인 기능은 메모리공간의 최상단에 놓여진 64k바이트의 ROM에 집약되어 있다. 여기에는 BIOS루틴 본체만이 아니고 ROM 베이직프로그램부분, 전원 ON시의 시스템 기능체크, 인터럽트벡터의 세트, I/O콘트롤러의 초기설정 등 리세트에서 IPL(Initial Program Loader)까지의 시스템상승에 필요한 모든 프로그램이 들어 있다.

또 특수용도의 옵션 어댑터를 위한 전용디바이스 드라이버는 보드차체가 인스톨되지 않을 가능성이 있으므로 어댑터 보드의 쪽에 실장하여 일반 BIOS와는 다른 영역을 확보하고 있다.

●BIOS의 기능

BIOS는 말한 것도 없이 기본 I/O 핸들러이므로 프로그래머를 I/O콘트롤러의 직접제어에서 해방하기 위한 서비스 루틴이라 생각하는 것이 타당한 것이다. 전자회로를 설계할 때에 IC

나 LSI의 기능에 관한 예비지식이 있고 없고는 완성되는 것이 아주 달라지는 것과 마찬가지로

- 어떤 기능이 있는가?
- 어떻게 하여 그 서비스를 받는가?
- 주의점이 있는가?

등에 대해서 알고 있는 것이 시스템을 이해하는데 도움이 될 것이다. BIOS루틴 그것의 내용에 관한 상세한 정보는 알 필요없다.

BIOS기능의 호출은 소프트웨어 인터럽트에 의해서 이루어진다.

〔표 1〕 IBM PC의 인터럽트기능 일람

00H	제로로 할당
01H	싱글 스텝
02H	NMI
03H	브레이크 포인트
04H	오버플로우
05H	프린트 스크린
08H	시스템 타이머
09H	키보드
0EH	디스켓
10H	비디오(디스 플레이)
11H	기기구성 정보
12H	메모리사이즈
13H	디스켓/하드 디스크
14H	RS-232C
15H	시스템 서비스
16H	키보드
17H	프린터
18H	ROM 베이직
19H	부트스트랩 로더
1AH	시스템 타이머/리얼타임 클럭
1BH	키보드 브레이크
1CH	유저 타이머 칩
1DH	비디오 파라미터
1EH	디스켓 파라미터
1FH	비디오 그래픽캐릭터
20H~3FH	DOS로 사용
40H	디스켓 리벡터
41H	하드디스크 파라미터
46H	하드디스크 파라미터
4AH	유저 얼라인
60H~67H	유저 프로그램으로 사용가능
70H	리얼타임 클럭
75H	NMI로의 리다이렉트
86H~F0H	베이직으로 사용
F1H~FFH	유저 프로그램으로 사용 가능

표 1에 IBM PC의 인터럽트 기능 일람을 나타낸다. 이들중 시스템이 이용하는 것은 유저 프로그램에서 콜하지 않는 것이 좋다. 시스템을 잘모를 경우 디스플레이, RS-232C, 프린터 등 극히 일반적인 I/O부분만 콜하는 것이 좋다.

●시스템 ID

처음나온 IBM PC시리즈나 최신의 PS/2 모두 BIOS의 기본기능은 거의 같다. 그러나 세밀하게는 조금씩 다르므로 주의가 필요하다. 즉 동일기종에서도 버전의 차이가 있으며, BIOS파라미터, 기능 등의 내용이 가지각색이다. 그러나 F000:FFFEH에 시스템ID라고 하는 것이 마련되어 있으므로 유저 프로그램측에서는 호환성의 불일치에 의한 불편함을 회피할 수 있다. 표 2에 시스템ID의 일람을 나타낸다. 통상 시스템ID의 취득방법은 BIOS-ROM을 직접 액세스하는 것은 아니고 BIOS콜(INT 15H)에 의해서 한다.

〔리스트 1〕 BIOS 콜링시퀀스의 예

```
mov ah,0   ;기능번호를 AH에 세트
int 16h    ;키보드 BIOS를 콜
            AH에 스캔코드가 들어간다.
            AL에 캐릭터 코드가
            들어간다.
```

●멀티태스크

대부분의 BIOS루틴은 응용 소프트웨어측에서 멀티태스크 동작을 하도록 BIOS를 콜하는 경우에는 주의가 필요하다. 현재의 시점에서는 디바이스 드라이버(BIOS)는 NETBIOS(옵션)만이며, 그 외는 직렬처리(배치처리적)밖에 접수하지 않는다. 따라서 다중콜은 아니고, 별도의 태스크가 BIOS를 사용하고 있는 경우에는 그 처리가 종료하기까지 기다리지 않으면 안된다(INT 15H 참조).

●BIOS의 호출방법

BIOS의 콜링 시퀀스는 필요한 파라미터가 있으면 이것을 각 레지스터에 세트하여 목적의 인터럽트를 걸 뿐이다. 만일 되돌아 오는 값이 있으면 제어가 되돌아 왔을 때에 소정의 레지스터에 격리되어 있다. MS-DOS의 펑션 콜과 방법은 같다.

예를 들면 키보드에서 1문자 입력하는 경우에는 리스트 1과 같이 된다. 이 리스트 1 예에서는 호출전의 파라미터의 수는 기능번호(AH)만으로 하나밖에 없지만 각 BIOS기능에 의해서 여러가지이다. 되돌아오는 값도 마찬가지이다.

중요한 BIOS들

●비디오에 관한 BIOS(INT 10H)

디스플레이관계는 꾸며 넣어진 디스

〔표 2〕 IBM PC 시리즈의 시스템 ID 일람표

기 종	BIOS일자	모델	서브모델	BIOS 개정
IBM PC	81/04/24	FFH	—	—
IBM PC	81/10/19	FFH	—	—
IBM PC	82/10/27	FFH	—	—
IBM PC-XT	82/11/08	FEH	—	—
IBM PC-AT	84/01/10	FCH	—	—
IBM PC-AT	85/06/10	FCH	00H	01H
IBM PC-AT	85/11/15	FCH	01H	00H
IBM PC-XT	86/01/10	FBH	00H	01H
IBM PC-XT/286	86/04/21	FCH	02H	00H
IBM PC-XT	86/05/09	FBH	00H	02H
PS/2 Model 30	86/09/02	FAH	00H	00H
PS/2 Model 50	—	FCH	04H	00H
PS/2 Model 60	—	FCH	05H	00H
PS/2 Model 80	—	F8H	00H or 01H	00H

플레이 어댑터에 의해서 서포트의 유무와 일부의 기능이 달라져 있다든지 하므로 주의가 필요하다. 표 3에 비디오관계의 기능일람을 나타낸다. PC-AT에서는 AH의 값 10H~12H가 예약, 14H~FFH가 미정이다.

표 4에 모드설정의 일람을 나타낸다. 모드의 종류는 많지만 해상도가 낮고, 그래픽은 EGA가 아니라도 충분하지 않을 것이다. 또 인스톨되어 있는 어댑의 종류에 의해서 해상도 등이 다르므로 프로그래밍시는 주의가 필요하다. PC-AT에서 사용할 수 있는 콜링 시퀀스에 대해서는 표 5에 나타내었다.

IBM PC시리즈에서 서포트되는 그래픽 · 캐릭터(폰트)는 표 6에 나타낸다.

●키보드에 관한 BIOS(INT 16H)

키보드는 기종과 버전에 따라서 하드웨어의 내용이 현저하게 달라진다. 그러나 BIOS를 경유함으로써 OS 혹은 어플리케이션 레벨에서의 키보드 출력코드의 호환성은 유지된다. 그러므로 기종에 의존하지 않는 프로그램을 쓰기 위해서도 BIOS는 중요한 역할을 한다. 표 7에 키보드 BIOS의 기능일람을 나타낸다. 그림 1과 같이 하면 버전 등을 체크할 수 있다.

각 BICS콜에 대해서는 표 8에 나타낸다. 그림중의 스캔·코드와 ASCII 코드에 대해서는 표 9에 나타낸다.

●RS-232C에 관한 BIOS(INT 14H)

표 10에 RS-232C 콘트롤기능의 일람을 나타낸다. 그리고 00H~03H까지의 기능에 대해서 표 11에 자세히 나타낸다.

●프린터에 관한 BIOS(INT 17H)

프린터에 관한 BIOS는 단지

〔표 3〕 디스플레이에 관한 BIOS의 기능 일람(INT 10H)

AH	기 능
00H	비디오 모드의 세트
01H	커서 타입의 세트
02H	커서 위치의 세트
03H	커서 위치의 리드
04H	라이드 펜 위치의 리드
05H	액티브 디스플레이 페이지의 선택
06H	액티브 페이지 위로 스크롤
07H	액티브 페이지 아래로 스크롤
08H	속성과 문자의 리드(커서 위치)
09H	속성과 문자의 라이트(커서위치)
0AH	문자의 라이트(커서위치)
0BH	컬러 팔레트의 세트
0CH	도트의 라이트
0DH	도트의 리드
0EH	액티브 페이지에 문자를 라이트(텔레타입)
0FH	현재의 비디오 상태 리드
10H	팔레트 레지스터의 세트
11H	캐릭터 제너레이터
12H	얼터네이트 셀렉트
13H	문자열의 라이트
14H, 15H	PC 호환 전용
16H~19H	예 약
1AH	디스플레이 조합 코드의 리드 또는 라이트
1BH	기능/상태 취득
1CH	상태 저장/처리장
1OH~FFH	예약

〔표 4〕 디스플레이 모드의 일람

모드	타입	컬러	텍스트	디스플레이	캐릭터	하드웨어 서포트	페이지
0, 1	텍스트	16	40×25	320×200 320×350 320×400 360×400	8×8 8×14 8×16 9×16	CGA,EGA,PS/2 EGA,PS/2 PS/2(M30) PS/2	8 8 8 8
2, 3	텍스트	16	80×25	640×200 640×200 640×350 640×400 720×400	8×8 8×8 8×14 8×16 9×16	CGA EGA,PS/2 EGA,PS/2 PS/2(M30) PS/2	4 8 8 8 8
4, 5	그래픽	4	40×25	320×200	8×8	CGA,EGA,PS/2*	1
6	그래픽	2	80×25	640×200	8×8	CGA,EGA,PS/2*	1
7	그래픽	모노크롬	80×25	720×350 720×350 720×400	9×14 9×14 9×16	MDPA EGA,PS/2 PS/2(M30)	1 8 8
D	그래픽	16	40×25	320×200	8×8	EGA,PS/2	8
E	그래픽	16	80×25	640×200	8×8	EGA,PS/2	4
F	그래픽	모노크롬	80×25	640×350	8×14	EGA,PS/2	2
10	그래픽	16	80×25	640×350	8×14	EGA,PS/2	2
11	그래픽	2	80×30	640×480	8×16	PS/2*	1
12	그래픽	16	80×30	640×480	8×16	PS/2	1
13	그래픽	256	40×25	320×200	8×8	PS/2*	1

콜 링 시 퀀 스	
입 력	AH : 00H AL : 비디오 모드
출 력	없음

MDPA : 모노크롬 어댑터
CGA : 컬러 그래픽 어댑터
EGA : 엔헨스트 그래픽 어댑터
PS/2(M30) : PS/2 모델 30
PS/2 : 모델 30 이외의 PS/2
PS/2* : 모든 PS/2

〔표 5〕 비디오 BIOS의 콜링시퀀스(INT 10H)

입 력	내 용	출 력	내 용
AH AL	00H…모드 설정 표 4 참조		없음
AH	01H…커서 타입 세트 도트라인(비트 4~0) 프린터(비트 6, 5) 모뎀라인(비트 4~0)		없음
AH BH DX	02H…커서위치 세트 페이지 번호 DH : 행, DL : 자리		없음
AH BH	03H…커서위치 리드 페이지 번호	DX CX	현재의 커서 위치, DH : 행, DL : 자리 커서 타입
AH	04H…라이트 펜위치의 리드	AX BX CX	00H…라이트 펜이 서포트되지 않는다 (BX, CX, DX는 변화한다). 01H…BX, CX, DX에 위치정보가 확정. 도트 컬럼위치(0~319, 639) 라인위치(0~199)
AH AL	05H…액티브 페이지에 선택 페이지 번호(표 4 참조)		없음
AH AL BH CX DX	06H…위로 스크롤 빈공간 있는 행수(00H : 빼낸 부분은 모두 빈공간) 블랭크 라인의 어트리뷰트 스크롤하는 좌측상부위치(CH : 행, CL : 자리) 스크롤하는 우측하부위치(DH : 행, DL : 자리)		없음
AH AL BH CX DX	07H…아래로 스크롤 블랭크하는 행수(00H : 빼낸 만큼 모두 블랭크) 블랭크 라인의 어트리뷰트 스크롤하는 좌측상부위치(CH : 행, CL : 자리) 스크롤하는 우측하부위치(DH : 행, DL : 자리)		없음
AH BH	08H…현재의 커서 위치의 속성/문자를 취득 페이지 번호	AH AL	속성(텍스트 모드만) 리드문자 코드
AH AL BL BH CX	09H…현재의 커서 위치의 속성/문자를 라이트 라이트 문자코드 문자속성(텍스트)/문자컬러(그래픽) 페이지 번호 라이트 문자의 수		없음
AH AL BH CX	0AH…현재의 커서 위치에 문자를 라이트 라이트 문자코드 페이지 번호 라이트 문자의 수		없음
AH BH BL	0BH…컬러 팔레트의 세트 00H…320×200 그래픽 모드 : 백 그라운드 컬러 텍스트 모드 : 보드 컬러 640×200 그래픽 모드 : 포 그라운드 컬러 01H…320×200 그래픽 모드 : 팔레트의 선택 0~31H…컬러 코드(BH가 00H일 때) 00H…그린(1)/레드(2)/브라운(3)(BH : 01H) 01H…시안(1)/마젠타(2)/화이트(3)(BH : 01H)		없음
AH AL BH CX DX	0CH…도트의 라이트 컬러 코드 페이지 번호(그래픽 모드시) 컬럼 번호 로우 번호		없음

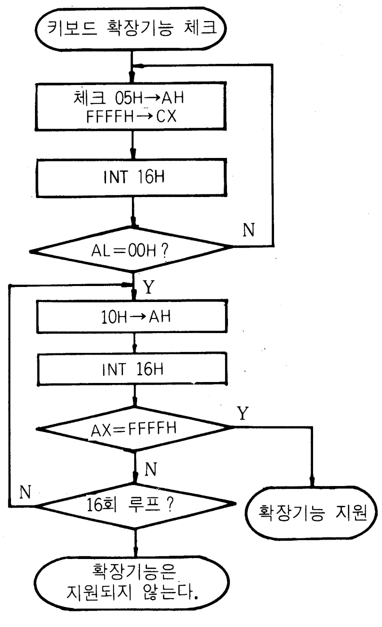
〔표 5〕 비디오 BIOS의 콜링 시퀀스(INT 10H)

입 력	내 용	출 력	내 용
AH	0DH…도트의 리드	AL	리드된 도트
BH	페이지 번호(그래픽 모드시)		
CX	컬럼 번호		
DX	로우 번호		
AH	0EH…액티브 페이지에 문자의 라이트(텔레타이프)		없음
AL	라이트 문자코드		
BL	그래픽 모드에 있어서 포그라운드 컬러		
AH	0FH…현재의 비디오 상태 취득	AH	컬럼 문자수
		AL	현재의 모드설정
AH	13H…문자열의 라이트		
	00H…BL : 문자속성, 단순한 문자열의 커서는 이동하지 않는다.		
	01H…BL : 문자속성, 단순한 문자열의 라이트 커서는 이동한다.		
	02H…문자 코드, 문자속성의 순으로 라이트 커서는 다동하지 않는다(텍스트 모드시).		
	03H…문자코드, 문자속성 순으로 라이트 커서는 이동한다(텍스트 모드시).		
BH	페이지 번호		
BL	AL=00H, 01H일 때 문자속성		
CX	문자수(속성은 포함되지 않는다)		
DX	커서 위치(문자열을 라이트하는 위치)		
ES:BP	문자열로의 포인터		

〔표 6〕 IBM PC의 그래픽 캐릭터

HEXA DECIMAL VALUE	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	BLANK (NULL)	►	BLANK (SPACE)	0	@	P	‘	p	Ç	É	á	░	└	╨	∝	≡
1	☺	◄	!	1	A	Q	a	q	ü	æ	í	▒	┴	╤	β	±
2	☻	↕	"	2	B	R	b	r	é	Æ	ó	▓	┬	╥	Γ	≥
3	♥	‼	#	3	C	S	c	s	â	ô	ú	│	├	╙	π	≤
4	♦	¶	$	4	D	T	d	t	ä	ö	ñ	┤	─	╘	Σ	⌠
5	♣	§	%	5	E	U	e	u	à	ò	Ñ	╡	┼	╒	σ	⌡
6	♠	▬	&	6	F	V	f	v	å	û	ª	╢	╞	╓	µ	÷
7	•	↨	’	7	G	W	g	w	ç	ù	º	╖	╟	╫	τ	≈
8	◘	↑	(	8	H	X	h	x	ê	ÿ	¿	╕	╚	╪	Φ	°
9	○	↓	)	9	I	Y	i	y	ë	Ö	⌐	╣	╔	┘	θ	•
A	◙	→	*	:	J	Z	j	z	è	Ü	¬	║	╩	┌	Ω	·
B	♂	←	+	;	K	[	k	{	ï	¢	½	╗	╦	█	δ	√
C	♀	∟	,	<	L	＼	l	¦	î	£	¼	╝	╠	▄	∞	n
D	♪	↔	—	=	M	]	m	}	ì	¥	¡	╜	═	▌	φ	2
E	♫	▲	.	>	N	∧	n	∼	Ä	₧	«	╛	╬	▐	∈	■
F	☼	▼	/	?	O	_	o	△	Å	ƒ	»	┐	╧	▀	∩	BLANK 'FF'

〔그림 1〕 키보드의 확장기능 체크방법



데이터를 프린터포트에 전송하기도 하고, 상태를 되돌리기도 한다. 그러나 상태 정보는 MS-DOS의 시스템 콜을 사용하는 것 보다도 상세한 내용을 얻을 수 있다. 표 12에 프린터BIOS의 기능을 나타낸다.

●디스크장치에 관한 BIOS(INT 13H)

하드디스크BIOS와 같은 인터럽트번호를 플로피 디스크BIOS에서도 사용하고 있다. 하드디스크와 디스켓의 구별을 드라이번호지정시의 비트7에 의해서 결정된다. 디스켓을 지정하는 경우는 비트7을 "0"으로 하지 않으면 안된다(하드디스크지정은 비트7을"1"로 한다. 따라서 하드디스크의 드라이브

번호는 80H이상부터가 된다).

BIOS루틴은 하드디스크, 플로피디스크 따로따로 쓰여져 있다. 기능적으로 같은 것은 같은 호출번호를 사용하지만 파라미터나 스테이터스 등은 다르다. 각 기능의 내용은 디스크의 리드/라이트는 중심으로 물리포맷, 각종 체크 등이 있다. 표 13에 플로피디스크와 하드디스크의 기능호출을 나타낸다.

통상은 MS-DOS 등의 OS상에서 동작시키는 것이 많으므로 이 펑션 콜은 특수한 응용시에 사용하는 것으로 될 것이다. 이 때 앞에서 설명한 바와 같이 BIOS루틴은 다중콜을 허용하지 않고 사용상황의 체크가 항상 필요하게 된다. 이것에 관해서는 시스템 서비스(INT 15H)를 참조하여야 한다.

표 14에 디스크 BIOS콜에 대해서 나타낸다.

●리얼타임 클럭에 관한 BIOS (INT 1AH)

시스템 날짜, 시간에 관한 조작은 모두 이 인터럽트 요구에 의해서 실행할 수 있다. 또 시스템 타이머의 세트, 알람기능도 포함하고 있다. 표 15에 BIOS의 기능콜을 나타낸다.

시스템 타이머 쪽은 8254에 입력되고 있는 클럭이 1.19318MHz이며, 이것을 65535분주하고 있으므로 1초당 18.2회의 인터럽트가 발생한다. 이 핸들러는 INT 08H이며, 전원 ON에서의 시간계측, 디스크 드라이브의 모터 OFF까지의 시간계측, 더 나아가서는 유저루틴 벡터가 세트되어 있으면 INT 1CH를 발생시켜서 타이머인터럽트 발생때마다 유저 루틴에 제어를 넘겨 준다.

리얼타임 클럭의 설정에 따라서는 하드웨어 인터럽트(IRQ8)를 실행하는 것이지만 인터럽트 핸들러루틴은 소프트인터럽트 INT 70H로 등록되어 있다. 이 기능은 주기적 인터럽트와 알람인터럽트의 콘트롤이다. 주기적 인터럽트는 INT 15H의 이벤트 웨이트와 웨이트, 알람인터럽트는 INT 1AH로 세트, 리세트된다. 알람인터럽트가 발생하면 INT 4AH가 발생한다.

●시스템 서비스에 관한 BIOS (INT 15H)

이 인터럽트는 I/O이외의 서비스도 포함하고 있다. 80286을 프로젝트모드로 이행시키기도 하고, 여러가지 잡다한 서비스도 수행하지만 시스템가동에 관한 중요한 기능콜이 있으므로 사용시에는 주의가 필요하다. 표 16에 BIOS의 기능콜일람을 나타낸다. 이렇게까지 많은 기능을 BIOS가 서포트하고 있으며, 더구나 이들 모두가 오픈되어 있으므로 어느 것으로 하여도 프로

〔표 7〕 키보드에 관한 BIOS의 기능 일람(INT 16H)

AH	기 능
00H	keyboard read
01H	keystroke status
02H	shift status
03H	set typematic rate
04H	keyboard click adjustment
05H	keyboard write
06H～0FH	reserved
10H	extended keyboard read
11H	extended keystroke status
12H	extended shift status
13H～FFH	reserved

〔표 8〕 키보드 BIOS의 콜링시퀀스

입력	내 용	출력	내 용
AH	00H…키보드 데이터의 리드	AH AL	스캔코드* ASCII코드*
AH	01H…키 눌린 상태	ZF AH AL	0 : AX에 데이터확립 1 : AX의 데이터는 사용되지 않는다. 스캔코드* ASCII코드*
AH	02H…시프트 상태	AH AL	예약되어 있다. 7 6 5 4 3 2 1 0 0 : Right Shift 키가 눌린 상태 1 : Left Shift 키가 눌린 상태 2 : Ctrl키가 눌린 상태 3 : Alt키가 눌린 상태 4 : Scroll Lock 상태 5 : Num Lock 상태 6 : Caps Lock 상태 7 : Insert Lock 상태

ZF : Zero Flag
*키보드 스캔 코드/캐릭터 코드 대응표 참조(표 9)

〔표 9〕 키보드 스캔코드/캐릭터코드 대응표

key	Esc	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12		SysRq	ScLck	Break				
	01/1B	3B/00	3C/00	3D/00	3E/00	3F/00	40/00	41/00	42/00	43/00	44/00										
shift	01/1B	54/00	55/00	56/00	57/00	58/00	59/00	5A/00	5B/00	5C/00	5D/00										
ctrl	01/1B	5E/00	5F/00	60/00	61/00	62/00	63/00	64/00	65/00	66/00	67/00										
alt		68/00	69/00	6A/00	6B/00	6C/00	6D/00	6E/00	6F/00	70/00	71/00										
key	`	1	2	3	4	5	6	7	8	9	0	-	=	BS	Insert	Home	Page Up	Num Lock	/	*	-
	29/60	02/31	03/32	04/33	05/34	06/35	07/36	08/37	09/38	0A/39	0B/30	0C/2D	0D/3D	0E/08	52/00	47/00	49/00	35/2F	37/2A	4A/2D	
shift	29/7E	02/21	03/40	04/23	05/24	06/25	07/5E	08/26	09/2A	0A/28	0B/29	0C/5F	0D/28	0E/08	52/00	47/00	49/00	35/2F	37/2A	4A/2D	
ctrl			03/00				07/1E					0C/1F		0E/7F		77/00	84/00				
alt		78/00	79/00	7A/00	7B/00	7C/00	7D/00	7E/00	7F/00	80/00	81/00	82/00	83/00								
key	Tab	Q	W	E	R	T	Y	U	I	O	P	[	]	\	Delete	End	Page Down	Home	↑	PgUp	+
	0F/09	10/71	11/77	12/65	13/72	14/74	15/79	16/75	17/69	18/6F	19/70	1A/5B	1B/5D	2B/5C	53/00	4F/00	51/00	47/00	48/00	49/00	4E/2B
shift	0F/00	10/51	11/57	12/45	13/52	14/54	15/59	16/55	17/49	18/4F	19/50	1A/7B	1B/7D	2B/7C	53/00	4F/00	51/00	47/37	48/38	49/39	4E/2B
ctrl		10/11	11/17	12/05	13/12	14/14	15/19	16/15	17/09	18/0F	19/10	1A/1B	1B/1D	2B/1C		75/00	76/00	77/00		84/00	
alt		10/00	11/00	12/00	13/00	14/00	15/00	16/00	17/00	18/00	19/00										
key	Caps Lock	A	S	D	F	G	H	J	K	L	;	'	Enter					←	(5)	→	
		1E/61	1F/73	20/64	21/66	22/67	23/68	24/6A	25/6B	26/6C	27/3B	28/27	1C/0D					4B/00		4D/00	
shift		1E/41	1F/53	20/44	21/46	22/47	23/48	24/4A	25/4B	26/4C	27/3A	28/22	1C/0D					4B/34	4C/35	4D/36	
ctrl		1E/01	1F/13	20/04	21/06	22/07	23/08	24/0A	25/0B	26/0C			1C/0A					73/00		74/00	
alt		1E/00	1F/00	20/00	21/00	22/00	23/00	24/00	25/00	26/00											
key	Shift	Z	X	C	V	B	N	M	,	.	/	Shift				↑		END	↓	PgDn	Enter
		2C/7A	2D/78	2E/63	2F/76	30/62	31/6E	32/6D	33/2C	34/2E	35/2F					48/00		4F/00	50/00	51/00	1C/0D
shift		2C/5A	2D/58	2E/43	2F/56	30/42	31/4E	32/4D	33/3C	34/3E	35/3F					48/00		4F/31	50/32	51/33	1C/0D
ctrl		2C/1A	2D/18	2E/03	2F/16	30/02	31/0E	32/0D										75/00		76/00	1C/0A
alt		2C/00	2D/00	2E/00	2F/00	30/00	31/00	32/00													
key	Ctrl		Alt	SPACE						Alt		Ctrl			←	↓	→	Ins		Del	
				39/20											4B/00	50/00	4D/00	52/00		53/00	
shift				39/20											4B/00	50/00	4D/00	52/30			
ctrl				39/20											73/00		74/00				
alt				39/20																	

〔표 10〕 RS-232C에 관한 BIOS의 기능 일람(INT 14H)

AH	기 능
00H	initialize the communication port
01H	send character
02H	recieve character
03H	read status
04H	extended initialize
05H	extended communications port control
06H～FFH	reserved

〔표 12〕 프린터 BIOS의 콜링시퀀스(INT 17H)

입력	내 용	출력	내 용
AH	00H…데이터 출력	AH	7 6 5 4 3 2 1 0 0 — Time Out 1 — Reserved 2 — Reserved 3 — I / O Error 4 — Selected 5 — Out of paper 6 — Acknowledge 7 — Not Busy
AH DX	01H…프린터포트의 초기화 프린터 포트(0, 1)	AH	동상
AH	02H…상태 리드 프린터 포트(0, 1)	AH	동상

〔표 11〕 RS-232C BIOS의 콜링시퀀스(INT 14H)

입 력	내 용	출 력	내 용
AH AL DX	00H…RS-232C 포트의 초기화 7 6 5 4 3 2 1 0 〔각 비트의 의미〕 비트 1, 0 워드 길이 (10 : 7비트, 11 : 8비트) 비트 2 스톱비트 (0 : 1스톱비트, 1 : 2스톱비트) 비트 4, 3 패리티 (X0 : 없음, 01 : 기수, 11 : 우수) 비트 7, 6, 5 보오레이트 (000 : 110bps, 001 : 150, 010 : 300, 011 : 600, 100 : 1200bps, 101 : 2400, 110 : 4800, 111 : 9600) 사용하는 포트번호(0, 1)	AH AL	7 6 5 4 3 2 1 0 〔라인 스테이터스〕 0 Data ready 1 Overrun error 2 Parity error 3 Framing error 4 Break detect 5 Transmitter holding register empty 6 Transmitter shift register empty 7 Time out 7 6 5 4 3 2 1 0 〔라인 스테이터스〕 0 Delta clear to send 1 Delta data set to ready 2 Trailing edge ring detector 3 Delta recieve line signal detect 4 Clear to send
AH DX	01H…데이터의 전송 사용하는 포토번호(0, 1)	AH AL	라인 스테이터스 입력 데이터가 그대로 보존된다.
AH DX	02H…데이터의 수신 사용하는 포트번호(0, 1)	AH AL	라인 스테이터스 수신한 데이터
AH DX	03H…상태 리드 사용하는 포트번호(0, 1)	AH AL	라인 스테이터스 모뎀 스테이터스

〔표 13〕 디스크 관계의 BIOS의 기능 일람(INT 13H)

AH	플로피 디 스 크 기 능	하드디스크 기능
00H	reset diskette system	reset disk system
01H	read status of last operation	read status of last operation
02H	read desired sectors into memory	read desired sectors into memory
03H	write desired sectors from memory	write desired sectors from memory
04H	verify desired track	verify desired sectors
05H	format desired track	format desired cylinder
06H	…	format desired cylinder and set bad sector flug
07H	…	format drive starting at desired cylinder
08H	read drive parameters	read drive parameters
09H	…	initialize drive pair characteristics
0AH~0BH	…	reserved for diagnostic
0CH	…	seek
0DH	…	alternate disk reset
0EH~0FH	…	reserved for diagnostic
10H	…	test drive ready
11H	…	recalibrate
12H~14H	…	reserved for diagnostics
15H	read disk type	read disk type
16H	diskette change line status	…
17H	set diskette drive type for format	…
18H	set media type for format	…
19H	…	park heads
1AH	…	format unit
1BH~FFH	reserved	reserved

그래밍하는 측에서는 큰 도움이 된다.

●주변기기의 구성정보를 얻는 BIOS (INT 11H)

시스템을 구성하고 있는 주변기기는 POST중에 조사되어 BIOS의 워크 에리어에 격납되어 있다. 이 인터럽트는 단지 그 정보를 AX에 넣어 돌려 주는 것이다. 응용 소프트웨어에서 기기정보가 필요한 경우에 콜하면 될 것이다. 예를 들면 80287이 인스톨되어 있는지 어떤지이며, 수치연산은 에뮬레이션하여 주는가, 직접80287의 명령을 내 놓는가 등 사용하기 쉬운 프로그램을 쓸 수 있다고 생각한다.

표 17에 AX의 각 비트의 의미를 나타낸다.

●메모리사이즈 정보를 얻는 BIOS (INT 12H)

사용가능한 메모리사이즈를 AX에 격납할 뿐이다. 이 값은 POST시에 체크된 것이다. 내용은 1024바이트 단위로 보고된다. 입력파라미터도 AX이외에는 없다.

BIOS사용상의 유의점

IBM PC시리즈에 마련되어 있는 BIOS는 상위호환성을 유지하도록 만들어져 있지만 반드시 그런것만은 아니다. 더구나 호환기종의 경우 기본적으로 중요한 것만을 지원하고 있다. 따라서 프로그램의 호환성을 확보하기 위해서는 OS가 가지고 있는 펑션은 될수록 OS에 대해서만 해야 한다.

MS-DOS하의 프로그래밍에서 BIOS콜이 필요로 하는 것은 비디오, 타이머, 시스템서비스 등이며, 그 외에 표준적인 I/O는 OS가 관리하고 있다. 따라서 유저 프로그램에서 직접 BIOS콜을 하며 BIOS용 워크에리어의 내용변경이 이루어지면 그후의 동작이 어떤 것으로 되는가는 보증의 한계가 아니다. 또 BIOS ROM은 버전업 서비스가 있

〔표 14〕 디스크 BIOS의 콜링시퀀스(INT 13H)

입력	내 용	출력	내 용
AH	00H…디스크 시스템의 리세트	CF	0 : 상태 정보 없음
DL	드라이브 번호(비트 7=0)		1 : 상태 정보 있음(장해검출)
		AH	80H : Diskette drive not ready
			40H : Seek operation failed
			20H : General controller failure
			10H : CRC error on diskette read
			0CH : Media type not found
			09H : Attempt to DMA across a 64kbyte boundary
			08H : DMA overrun on operation
			06H : Diskette change line active
			04H : Requested sector not found
			03H : Write protect error
			02H : Address mark not found
			00H : No error
AH	01H…직전의 동작/상태 리드	CF	동상
DL	드라이버 번호(비트 7=0)	AH	동상
AH	02H…섹터→메모리 리드	CF	동상
AL	리드 섹터수	AH	동상
ES : BX	버퍼 어드레스	AL	전송되는 섹터수
CH	트랙 번호		
CL	섹터 번호		
DH	헤드 번호		
DL	드라이브 번호(비트 7=0)		
AH	03H…메모리→섹터 라이트	CF	동상
AL	라이트 섹터수	AH	동상
ES : BX	버퍼 어드레스	AL	전송된 섹터수
CH	트랙 번호		
CL	섹터 번호		
DH	헤드 번호		
DL	드라이브 번호(비트 7=0)		
AH	04H…섹터리드 패리티	CH	동상
AL	패리티 섹터수	AH	동상
ES : BX	버퍼 어드레스	AL	패리티 섹터수
CH	트랙 번호		
CL	섹터 번호	CF	
DH	헤드 번호		
DL	드라이브 번호(비트 7=0)		
AH	05H…트랙 포맷	CF	동상
AL	포맷하는 트랙수	AH	동상
CH	트랙 번호		
DH	헤드 번호		
DL	드라이브 번호(비트 7=0)		
ES : BX	버퍼 어드레스(4바이트의 버퍼가 필요)		

〔표 14〕 디스크 BIOS의 콜링 시퀀스(INT 13H)

입력	내 용	출력	내 용
AH DL	15H…디스크 타입의 리드 드라이브 번호(비트 7=0)	AH CF	00H : 디스크 드라이브가 없다. 01H : 디스켓 교환 검출이 되지 않는다. 02H : 디스켓 교환 검출 가능 03H : 하드 디스크를 위한 예약 0 : 정상종료 1 : AH의 정보 불확정
AH DL	16H…디스켓 교환 라인 스테이터스 드라이브 번호(비트 7=0)	AH CF 캐리 플래그	00H : 디스크는 교환되지 않는다. 01H : 디스켓 파라미터가 불확정 06H : 디스켓이 교환된다. 80H : 드라이브가 레디상태가 아니다. 0 : AH=0이다. 1 : 디스켓이 교환되거나 또는

는 것도 아니므로 새로운 I / O에 대한 서포트는 낡은 ROM-BIOS에서는 기대할 수 없다. 그러나 마우스 등과 같이 응용 소프트웨어 쪽에서는 필요한 경우 MS-DOS의 디바이스 드라이버의 형태로 공급되고 있다.

MS-DOS의 특징

●MS-DOS에 대해서

BIOS가 직접 하드웨어를 제어하는 것과는 달리 MS-DOS는 I / O 콘트

〔표 15〕 시스템 클럭/리얼타임 클럭 BIOS의 콜링시퀀스(INT 1AH)

입력	내 용	출력	내 용
AH	00H…시스템 타이머의 타임카운터의 리드 (8254의 채널0)	AL CX DX	0 : 시스템 리세트 또는 카운터 리드, 세트로부터 24시간을 초과하지 않는다. 1 : 초과된다. 상위 카운트 하위 카운트
AH CX DX	01H…시스템에 타이머의 타임 카운터의 설정 상위 카운트 하위 카운트		없음
AH	02H…리얼타임 클럭의 리드 (MC146818)	CF CH CL DH DL(*1)	0 : 정상종료 시(BCD) 분(BCD) 초(BCD) 0 : 여름시간(*2) 옵션 OFF 1 : 여름시간(*2) 옵션 ON
AH CH CL DH DL	04H…리얼타임 클럭의 설정 시(BCD) 분(BCD) 초(BCD) 0 : 여름시간(*2) 옵션 OFF 1 : 여름시간(*2) 옵션 ON		없음
AH	04H…리얼타임 클럭의 리드	CF CH CL DH	0 : 정상종료, 1 : 클럭이 동작하지 않는다. 서기 상위 2자리(*3) (BCD 19 또는 20) 서기 하위 2자리(BCD) 월(BCD) 일(BCD)
AH DH	05H…리얼타임 클럭의 설정 서기 상위 2자리(*3)(BCD 19 또는 20) 서기 하위 2자리(BCD) 월(BCD) 일(BCD)		없음
AH CH CL DH	06H…리얼타임 클럭 얼라인의 설정 시(BCD) 분(BCD) 초(BCD)	CF	0 : 정상종료 1 : 얼라인이 세트되면 클럭이 동작하지 않는다.

(*1) 초기의 AT에는 지원되지 않았다. (*2) 여름시간 개시 : 4월 마지막 일요일 AM 1 : 59 : 59를 AM 3 : 00 : 00로 한다.
여름시간 종료 : 10월 마지막 일요일 AM 1 : 59 : 59를 AM 1 : 00 : 00로 한다.
(*3) BIOS 리스트에서는 19~20으로 되어 있지만 이 값은 단지 CMOS RAM의 어드레스 32H에 내장되는 것 뿐이다.

〔표 16〕 시스템 서비스에 관한 BIOS의 콜링시퀀스(INT 15H)

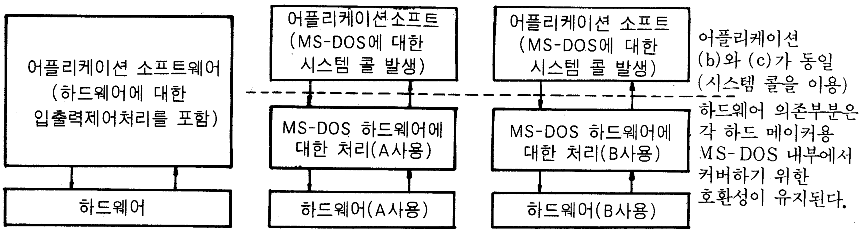
입력	내 용	출력	내 용
AH BX CX	80H…디바이스 오픈 디바이스 ID 프로세서 ID		
AH BX CX	81H…디바이스 클로즈 디바이스 ID 프로세서 ID		
AH BX	82H…프로그램 종료 디바이스 ID		
AH AL ES : BX CX,DX	83H…이벤트 웨이트 00H…세트 간격 (ES : BX, CX, DX) 01H…리세트 간격 인터벌 값	CF	0 : 정상종료 1 : 에러
AH DX DX	84H… 스티닝 서포트 00H… 현재의 스위치 설정을 읽어 들임 01H… 좌표입력을 읽어 들임	CF AL AX BX CX DX	1 : 에러 B2, B1, A2, A1, 0, 0, 0, 0 A의 X좌표 A의 Y좌표 B의 X좌표 B의 Y좌표
AH AL	85H… SysReq키를 누른다. 00H… SysReq키를 누른다. 01H… SysReq키를 놓는다.		
AH CX DX	86H…웨이트 웨이트 값 웨이트 값	CF	0 : 정상종료 1 : 설정종류(에러)
AH CX ES : SI	87H…메모리 내용의 블록전송 워드단위의 전송 사이즈(최대 32k워드) GDT(Gloval Descriptor Table) 어드레스[*1]	AH	00H : 정상종료 CF=0 ZF=1 01H : RAM 패리티 에러 02H : 다른 예외처리 에러 CF=0 ZF=1 03H : 어드레스(A20) 라인 불량
AH	88H…확장 메모리 사이즈정보	AX	리얼모드 이외(1MB이상)의 1KB단위의 메모리 사이즈
AH ES : SI	89H…프로텍트 모드이행 GD 어드레스 [*2]	AH	00H에서 정상종료 모든 세그먼트 레지스터가 변화한다. AX, BP는 변경된다.
AH AL	90H…디바이스 비지 타입 코드 00H : 하드 디스크 01H : 디스켓 02H : 키보드 03H : 포인팅 디바이스 80H : 네트워크 (ES : BX) : 네트워크 콘트롤 블록 FDH : 디스크 모터 스타트 FEH : 프린터	CF	0 : 웨이트 없음 1 : 요소 디바이스 타입에 웨이트 타임있음
AH AL	91H… 인터럽트 완료 타입코드 00H : 하드 디스크 01H : 디스켓 02H : 키보드		

입력	내 용	출력	내 용
	03H : 포인팅 디바이스 08H : 네트워크 (ES : BX) 네트워크 콘트롤 블록 FDH :디스크 모터 스타트 FEH :프린터		
AH	C0H …시스템 구성정보의 취득	AH CF ES : BX	00H 00H 시스템 디스크립터로의 포인터 DW 데이터수(바이트 단위) DB 모델 DB 서브 모델 DB BIOS 버전 번호 DB 하드 / 소프트 구성 7 6 5 4 3 2 1 0 0: 예약 1: 1 : 마이크로채널 0 : PC타입 I / O채널 2: 확장 BIOS 1 : 있음 0 : 없음 3: 외부 인벤트용의 웨이트 1 : 있음 0 : 없음 4: 키보드 인터럽트(INT 15H) 콜 1 : 있음 0 : 없음 5: 리얼타임 클럭 1 : 있음 0 : 없음 6: 슬레이브 8259A 1 : 있음 0 : 없음 7: DMA 채널 3 1 : 고정 디바이스가 사용됨 0 : 미사용을 사용할 수 없음 DB : 예약 DB : 예약 DB : 예약 DB : 예약

(*1) 블록 전송용 GDT
ES : SI+0 더미
+8 GDT 로케이션
+10 소스 GDT
+18 타켓 GDT
+20 BIOS 코드 세그먼트(CS)
+28 BIOS 소스 세그먼트(CS)

(*2) 프로텍트 모드 이행용 GDT
ES.: SI+0 더미
+8 GDT
+10 IDT(Interrupt Descriptor Table)
+18 DS
+20 ES
+28 SS
+30 CS
+38 BIOS CS

〔그림 2〕 소프트웨어의 호환성



〔표 17〕 기기 구성 정보의 콜링시퀀스(INT 11H)

입력	내 용	출력	내 용
	없음	AX	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 (12, 8, 3 ■은 미사용) 0: IPL 디스켓 1: 수치연산 프로세서 2: 포인팅 디바이스 4–5: 비디오 모드 타입 — 00…예약 / 01…40×25 (컬러) / 10…80×25 (컬러) / 11…80×25 (모노크롬) 6–7: 디스크드라이브의 수 — 00…1드라이브 / 01…2드라이브 9–11: RS-232C카드의 수 13: 내장 모뎀 14–15: 프린터의 접속수

롤만을 행한다. 따라서 MS-DOS는 이 기종간에서의 소프트웨어의 호환성이나 프로그램개발 등의 충실이라고 하는 것을 목적으로 탄생하였다.

예를 들면 A사의 하드웨어용 소프트웨어를 B사의 하드웨어에서 사용할 수 없는 것은 자주 있는 것이며, 이것은 A사의 하드웨어에 의존한 부분이 있기 때문이다.

MS-DOS는 이와 같은 하드웨어 구성의 차이에서 일어나는 문제를 해소하는 것이 그 목적이다. 즉 MS-DOS를 사용하며 하드웨어에 의존한 소프트웨어사양을 흡수시킴으로써 하드웨어와 응용 소프트웨어의 완충지대로 되어 하드웨어에 의존하지 않는 소프트웨어를 작성할 수 있다.

이것을 간단히 설명하면 **그림 2**(a)와 같이 MS-DOS가 없으면 하드웨어에 의존하는 부분도 응용 소프트웨어 내부에 가지고 있지 않으면 안된다. 그러나 그림 2(b), (c)와 같이 MS-DOS를 사용하면 하드웨어에 의존하는 부분은 MS-DOS에 의해서 흡수되고 있으므로 응용소프트웨어 내부에서는 MS-DOS에 대한 공통의 명령(시스템 · 콜)을 사용함으로써 응용 소프트웨어 호환성은 유지된다.

●MS-DOS의 구조

MS-DOS는 3개의 부분으로 나누어지고 있다. 그리고 그 각 부분이 서로 연관되어 OS의 동작을 하고 있다.

이들을 COMMAND.COM, MSDOS.SYS, IO.SYS로 구성되어 있다.

COMMAND.COM은 유저의 코맨드를 접수, 그 의미를 해석하여 실행으로 옮기는 부분이며, 유저와 컴퓨터의 인터페이스 프로그램이다. 그 외에 MS-DOS의 내부코맨드라고 불리우는 부분은 이 속에 꾸며 넣어져 있다.

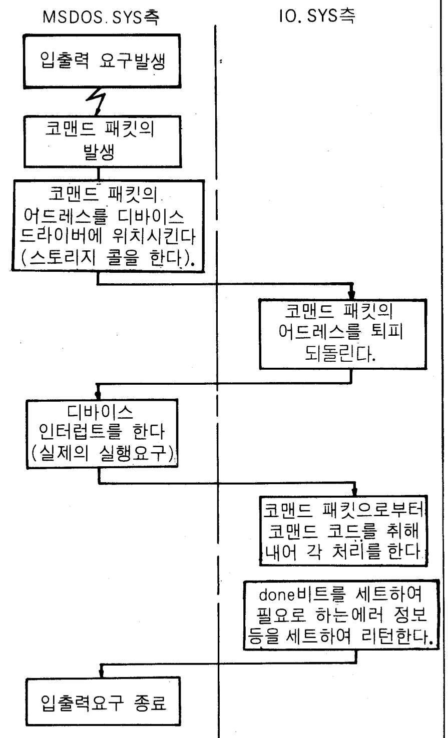
MSDOS.SYS는 MS-DOS의 본체이며, IO.SYS에 의한 주변장치의 입출력기능을 사용하여 콘솔의 입출력이나 디스크를 관리하는 화일시스템 등을 실현한다. 즉 유저소프트웨어와 IO.SYS간의 인터페이스적인 처리를 한다.

IO.SYS는 주변장치(디바이스)의 하드웨어를 직접조작하는 저레벨(하드웨어에 가까울수록 실제로는 IO.SYS가 직접조작하는 것은 적고, 퍼스널컴퓨터에 꾸며 넣어져 있는 ROM-BIOS (Basic Input / Output System)를 사용함으로써 이루어지고 있다. 또 IO.SYS의 내부에 있으며 각 주변장치를 제어하는 프로그램을 디바이스드라이버라고 한다.

●MSDOS.SYS와 IO.SYS

응용 소프트웨어의 실행중에 주변장치에 대한 입출력처리의 요구를 받은 MSDOS.SYS는 어느 디바이스드라이버에 대해서 어떤 액세스를 하면 되는가를 판단하여 목적의 디바이스드라이버에 코맨드에 보낸다. 필요하다면 그 주변장치의 상태를 조사하여 입출력이 가능한가, 상태가 변화하였는가 등을 SYS가 출력하는 코맨드에 따라서 주변장치의 상태를 조사하기도 하고, 데이터의 입출력을 하기도 하며 그 결과를 MSDOS.SYS의 되돌리는 등 면밀한 관계로 되어 있다.

〔그림 3〕 디바이스 드라이버의 호출방법



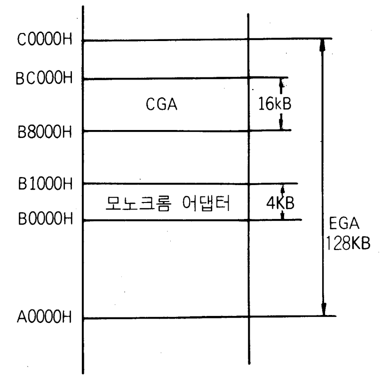
●디바이스드라이버의 호출법

그림 3에 나타낸 바와 같이 MSDOS.SYS는 우선 처음에 디바이스드라이버와의 사이에서 코맨드나 데이터를 주고받기 위해서 사용하는 코맨드 패킷의 포인터를 얻는 동작을 한다. 즉 MS-DOS.SYS에서는 코맨드 · 패킷의 선

〔표 18〕 I/O 어드레스

I/O 어드레스	IBM PC
000H~01FH	$DMAC_1$ (8237A)
020H~03FH	8259A마스터
040H~05FH	8254
060H~06FH	8042(Key I/F)
070H~07FH	캘린더
080H~09FH	DMA Page Reg.
0A0H~0BFH	8259A 슬레이브
0C0H~0DFH	$DMAC_2$ (8237A)
1F0H~1F8H	하드디스크
278H~27FH	프린터 포트 2
2F8H~2FFH	RS-$232C_2$ (8250)
378H~37FH	프린터 포트 1
3B0H~3BFH	모노크롬 디스플레이 I/F
3C0H~3CFH	EGA
3D0H~3DFH	CGA
3F0H~3F7H	FD I/F
3F8H~3FFH	RS-$232C_1$ (8250)

〔그림 4〕 CRT 표시용 VRAM 메모리맵



〔표 19〕 인터럽트 벡터의 차이점

INT NO.(H)	IBM PC	
0	제산에러	
1	싱글스텝	
2	NMI	
3	INT_3	
4	오버 플로우	
5	Print Screen	
6	시스템 예약	
7	시스템 예약	
8	Time of Day (TIMER_INT)	Ⓜ
9	Keyboard (KB_INT)	Ⓜ
A	시스템 예약	
B	RS-232C	Ⓜ
C	RS-232C	Ⓜ
D	Alternate Printer	Ⓜ
E	Diskette (DISK_INT)	Ⓜ
F	프린터	Ⓜ
10	Video (VIDEO_IO)	ⓑ
11	Equipment Check	ⓑ
12	Memory (MEMORY_SIZE_DETERMINE)	ⓑ
13	Diskette/Disk (DISKETTE_IO)	ⓑ
14	Communicatons (RS232_IO)	ⓑ
15	Cassette (CASSETTE)	ⓑ
16	키보드 (KEYBOARD_IO)	ⓑ
17	프린터 (PRINTER_IO)	ⓑ
18	ROM BASIC	
19	Bootstrap (BOOTSTRAP)	ⓑ

INT NO.(H)	IBM PC	
1A	Time of Day (TIME_OF_DAY)	ⓑ
1B	Keyboard Break	ⓑ
1C	Timer Tick	ⓑ
1D	Video Initialization	ⓑ
1E	Diskette Parameter (DISK_BASE)	ⓑ
1F	Video Graphics Chars	ⓑ
20~27	MS-DOS 펑션	
28~3F	MS-DOS 예약	
40~5F	BIOS 예약	
60~67	유저용	
68~6F	미사용	
70	IRQ_8 리얼타임 클럭	Ⓢ
71	IRQ_9 리다이렉트	Ⓢ
72	IRQ_{10}	Ⓢ
73	IRQ_{11}	Ⓢ
74	IRQ_{12}	Ⓢ
75	IRQ_{13} 80287	Ⓢ
76	IRQ_{14}	Ⓢ
77	IRQ_{15}	Ⓢ
78~7F	미사용	
80~85	BASIC	
86~F0	유저 (BASIC)	
F1~FF	미사용	

Ⓜ은 마스크 8259A를 나타낸다
Ⓢ는 슬레이브 8258A를 나타낸다.
ⓑ는 BIOS를 나타낸다.

두어드레스를 세이브하여 리턴한다. 그리고 다시 MSDOS.SYS는 코맨드패킷에 파라미터를 세트하고, 디바이스 인터럽트를 한다.

●코맨드 패킷

BIOS를 콜할 경우는 AH레지스터에 콜하는 펑션 파라미터를 세트한다. 그 외에 데이터, 테이블 등의 어드레스는 각종 레지스터를 사용하여 넘겨 준다. 그리고 표에 표시되는 각각의 레벨의 소프트웨어 인터럽트를 실행함으로써 BIOS를 콜하는 것으로 된다.

① Video BIOS

IBM PC에서는 캐릭터・제너레이터를 가지고 있지 않으므로 ROM에서 캐릭터 폰트데이터를 읽어 들여서 VRAM에 도트이미지로 써넣고 있다.

또 어댑터보드로서 모노크롬 컬러 그래픽(CGA), 엔헌스트・컬러・그래픽(EGA) 등이 있으며, 각각 I/O 및 메모리 맵이 다르다(그림 4, 표 18 참조).

② DISK BIOS

IBM PC에서는 디스크의 미디어 타입에 의한 코맨드의 차이는 없다. 현재 접속되어 있는 플로피디스크 드라이브에 어떤 타입의 미디어가 삽입되어 있는가는 메모리의 절대어드레스 40:90H(드라이브 A)와 40:91H(드라이브 B)에 격납되어 있는 값에서 알 수 있다.

③ RS-232C BIOS

IBM PC에서는 RS-232C의 콘트롤러에 8250A를 사용하고 있으므로 보오레이트용의 클럭분주가 콘트롤러에 내장되어 있다. 또 모뎀 스테이터스 콘트롤도 모두 내장되어 있다.

④ 키보드BIOS

IBM PC에서는 좌우의 시프트키가 구별되며 IBM PC에서의 시프트 스테이터스 키는 Shift, Ctrl, Alt, CapsLock, ScrollLock, NumLock가 있다.

●인터럽트벡터

표 19에 인터럽트벡터의 일람을 나타낸다.

IBM PC의 경우 코프로세서(80287)의 인터럽트가 발생하면 인터럽트처리 종류후 NMI(INT 02H 명령을 실행)로 점프한다.

●I/O 어드레스

표 18에 나타낸 IBM PC의 I/O어드레스를 보면 IBM PC에서는 I/O 어드레스의 A_0~A_9를 풀디코드하고 있다. 어드레스 0000H~00FFH까지가 본체에 장비되어 있는 I/O이며, 0100H부터는 확장슬롯에 장비되어 있는 보드의 I/O어드레스이다. PC

THE PROGRAMMER'S SHOP

helps save time, money, and cut frustrations. Compare, evaluate, and find products.

RECENT DISCOVERY

C Talk by CNS - Object-oriented C programming. Smalltalk-like message formats contains encapsulation, messaging, inheritance. MS, Lattice, Turbo, C86. MS $ 129

386 Development Tools

386 Assembler/Linker	PC	$ 379
386 Debug - by Phar Lap	PC	$ 139
386/DOS Extender	PC	$1395
DESQview PS/2	PC	$ 109
F77–LEM/32 bit – by Lahey	MS	$ 869
OS/286 & 386 by AI Architects	PC	Call
Paradox 386	MS	$ 799
PC-MOS/386	PC	$ 169
Windows/386	MS	Call

AI Expert System Dev't

1st Class	MS	$ 399
Arity Combination Package	PC	$ 979
System - use with C	MS	$ 229
CxPERT V3.0 - shell for C	MS	$ 349
Experteach II	PC	$ 339
Exsys	PC	$ 289
Level 5 - formerly Insight 2	MS	$ 589
T.I.: PC Easy	PC	$ 419
Personal Consultant Plus	PC	$2589
Turbo Expert-Startup(400 rules)	PC	$ 119
Corporate (4000 rules)	PC	$ 339

AI-Lisp

PC Scheme LISP - by TI	PC	$ 77
Star Sapphire	MS	$ 429
TransLISP - learn fast	MS	$ 79
TransLISP PLUS	MS	$ 139

Others: IQ LISP ($155), IQC LISP ($269)

AI-Prolog

APT - Active Prolog Tutor	PC	$ 49
ARITY Prolog - Interpreter	PC	$ 229
COMPILER/Interpreter-EXE	PC	$ 569
Cogent Prolog Compiler	MS	$ 179
MicroProlog Prof. Comp./Inter.	MS	$ 609
TURBO PROLOG by Borland	PC	$ 69
Turbo Prolog Toolbox	PC	$ 69

Basic

BAS_C - economy	MS	$ 179
BAS_PAS - economy	MS	$ 135
Basic Development System	PC	$ 105
Basic Development Tools	PC	$ 79
db/Lib	PC	$ 119
Exim Toolkit - full	PC	$ 85
Finally - by Komputerwerk	PC	$ 85
Inside Track	PC	$ 45
Mach 2 by MicroHelp	PC	$ 55
NetWurkz by Exim	PC	$ 89
Peeks 'n Pokes	PC	$ 30
QBase	MS	$ 79
QuickBASIC	MS	$ 63

FEATURES

CO/SESSION - Remotely: Access PC, peripherals, train or trouble-shoot. Session record/playback, file transfer keyboard and voice modes MS $ 229

CLARION DBMS by Barrington Systems. Fast applications prototyping and development. Language, compiler, screen/ report generators, editor, call other languages, read/write dBASEIII + files. PC Call

LIFEBOAT INC.
678-39, Yuksam-Dong, Kangnam-Ku, Seoul, Korea
PHONE : (02)555-1392
FAX : (02)555-9250

Basic

Quick Pak-by Crescent Software	PC	$ 59
Quick-Tools by BC Associates	PC	$ 109
True BASIC	PC	$ 79
Turbo BASIC - by Borland	PC	$ 69
Turbo BASIC Database Toolbox	MS	$ 69

C Language-Compilers

AZTEC C86 - Commercial	PC	$ 499
C86 PLUS - by CI	MS	$ 359
Datalight Optimum-C	MS	$ 109
High C Optimizing Compiler	PC	Call
Instant-C - source, debug.	MS	$ 369
Instant-C/16M	PC	Call
Lattice C – new version	MS	$ 249
Microsoft C 5.0 - Codeview	MS	$ 299
Microsoft Quick C	MS	$ 69
Rex-C/86-standalone ROM	MS	$ 695
Turbo C by Borland	PC	$ 64

C Language-Interpreters

C-terp by Gimpel - full K & R	MS	$ 219
C Trainer - by Catalytix	PC	$ 89
Interactive C by IMPACC Assoc.	PC	$ 189
Run/C Professional	MS	$ 145
Run/C Lite	MS	$ 79
Turbo C-terp	PC	$ 119

C Libraries-Communications

Asynch Manager by Blaise	PC	$ 135
Essential Comm Library	PC	$ 125
With Breakout Debugger	PC	$ 189
Greenleaf Comm Library	PC	$ 129
Multi-Comm - add multitasking	PC	$ 135

C Libraries-General

Blackstar C Function Library	PC	$ 89
C Tools Plus - V5.0	PC	$ 99
C Utilities by Essential	PC	$ 119
Entelekon C Function Library	PC	$ 119
Greenleaf C Sampler	PC	$ 69
Greenleaf Functions	PC	$ 125
LIGHT TOOLS by Blaise	PC	$ 69
Turbo C Tools by Blaise	PC	$ 99

C Libraries-Files

BTree - Source, no royalties	MS	$ 69
CBTREE-Source, no royalties	MS	$ 129
ctree by Faircom-no royalties	MS	$ 309
ctree w/ rtree	MS	$ 519
rtree - report generation	PC	$ 239
dB2C Files	MS	$ 259
dB2C Toolkit V2.0	MS	$ 249
dbVISTA - Object only	MS	Call

RECENT DISCOVERY

DBug III by HJS Research - Source level Dbase debugger. Breakpoints, view variable values, stop execution when variable changes, multiple windows. Stop runaway programs. MS $ 159

C-Screens, Windows, Graphics

C Power Windows by Entelekon	PC	$ 129
C Worthy Interface Library	PC	$ 165
Curses by Aspen Scientific	PC	$ 109
dBASE Graphics for C	PC	$ 69
ESSENTIAL GRAPHICS-fast	PC	$ 185
Font WINDOW/PLUS	PC	$ 229
GraphiC - new color version	PC	$ 279
Greenleaf Data Windows	PC	$ 155
w/source	PC	$ 259
LightWINDOWS/C for Datalight	PC	$ 79
Panel/TC - for Turbo C	PC	$ 95
PC Forms-by Golden	MS	$ 119
ScreenStar - with source	PC	$ 169
Terminal Mapping System	PC	$ 279
TurboWINDOW/C-for Turbo C	PC	$ 75
View Manager - by Blaise	PC	$ 199
Vitamin C - source, menus	PC	$ 155
VC Screen - screen paint	PC	$ 129
Windows for C - fast	PC	Call
Windows for Data - validation	PC	Call
ZView - screen generator	MS	$ 149

C Support-Systems

Advantage C + +	PC	$ 479
cENGLISH SK	MS	$ 89
C Sharp - **realtime**, tasks.	PC	$ 479
C ToolSet DIFF, xref, source	MS	$ 79
Lattice Text Utilities	MS	$ 89
Multi-C - multitasking	PC	$ 135
PC LINT-Checker, Amiga $89	MS	$ 99
PforCe + +	PC	$ 219
Pre-C - Lint-Like	MS	$ 169
Quickshell - script compiler	PC	$ 349
/*resident-C*/ with source	PC	$ 159
Sapiens V8 - 8M workspace	PC	$ 269
SECURITY LIB - Source $199	PC	$ 115
Timeslicer	PC	$ 265

Cobol

Cobol spII by Flexus	PC	$ 329
Microsoft COBOL	MS	Call
PCDT by Procode	MS	$ 899
Realia - very fast	MS	$ 799
RM/Screens	PC	$ 339
RM/User + 5	MS	Call
Ryan McFarland COBOL	MS	$ 899
COBOL-85	MS	$ 899

DataBase & File Management

Advanced Revelation	PC	$ 779
CQL - Query System - source	PC	$ 359
DataFlex by Data Access	PC	$ 595
DataFlex multiuser	PC	$1049
Magic PC	PC	$ 169
Paradox - original	PC	$ 369
Paradox V2.0 List: $725	PC	$ 499

FEATURE

CLEAR by Clear Software - Clarify Logic for Enhancement, Analysis, and Review. Graphical flow of control for dBASE. HP $135 MS $ 89

We support MSDOS (not just compatibles), PCDOS, Xenix-86, CPM-80, Macintosh, Atari ST, and Amiga.

THE PROGRAMMER'S SHOP

provides complete information, advice, guarantees, and every product for Microcomputer Programming.

Current Prices of Popular Products.

DBASE Language

Clipper compiler	PC	$ 379
dBASE III Plus	PC	$ 399
dBASE III LANPack	PC	$ 649
DBXL Interpreter	PC	$ 99
FoxBASE+ - V2.1	MS	Call
Quicksilver Diamond	PC	$ 369

DBASE Support

dAnalyst	PC	$219
dBASE Tools for C	PC	$ 65
dBRIEF with BRIEF-Auto-Indent, views structures	PC	Call
dBC III by Lattice	MS	$169
dBC III Plus	PC	$509
dBug III	MS	$159
Dialect UI	PC	$ 45
dFLOW - flow charts	MS	$125
Documentor - dFlow superset	MS	$229
FORCE III	PC	$109
Genifer by Bytel-code generator	MS	$279
QuickCode III Plus	MS	$189
R&R Report Writer	MS	$139
Seek-It - Query-by-example	PC	$ 79
Silver Comm Library	MS	$135
Tom Rettig's Library	PC	$ 79
UI Programmer-user interfaces	PC	$249

Debuggers

Breakout - by Essential	PC	$ 89
CODESMITH - visual	PC	$ 99
C SPRITE - data structures	PC	$119
DSD87	PC	$ 99
Periscope I	PC	$275
Periscope II	PC	$139
Periscope II-X	PC	$105
Periscope III-10 MHZ Version	PC	$795
Pfix-86 Plus - by Phoenix	PC	$209
SoftProbe II - embedded systems	PC	$695

Editors for Programming

BRIEF Programmer's Editor	PC	Call
de - EMACS-style	PC	$ 65
EMACS by UniPress Source:	$895	$265
Epsilon - like EMACS	PC	$149
KEDIT - like XEDIT	PC	$ 99
ME Macro Editor - Source	PC	$ 79
MKS VI	MS	$ 65
PC/EDT - macros	PC	$229
PC/VI - by Custom Software	MS	$109
Personal REXX - V1.6	PC	$ 99
SPF/PC - Version 2.0	PC	$179
Vedit PLUS	MS	$129

Note: All prices subject to change without notice. Mention this ad. Some prices are specials. Ask about COD and POs. Formats: 3" laptop now available, plus 200 others. UPS surface shipping add $3 item.

Fortran & Supporting

ACS Time Series	MS	$399
Forlib+	PC	$ 55
Fortran Addenda	PC	$139
I/O Pro - includes No Limit	PC	$229
MACFortran by Microsoft	MS	$229
MS Fortran - 4.0, full 77'	MS	$295
PC-Fortran Tools - xref, pprint	PC	$165
RM/Fortran	MS	$399
Scientific Subroutines - Matrix	MS	$129

Multilanguage Support

BTRIEVE ISAM	MS	$185
BTRIEVE/N - multiuser	MS	$455
GSS Graphics Dev't Toolkit	PC	$375
Halo Development Package	MS	$389
HALO Graphics	PC	$209
Help/Control - on line help	PC	$ 99
HOOPS 3D Graphics Library	PC	$399
Informix 4GL-application builder	PC	Call
Informix SQL - ANSI standard	PC	Call
Instant Programmer's Help	MS	$ 79
NET-TOOLS - NET-BIOS	PC	$129
Norton Guides	PC	$ 75
Opt Tech Sort - sort, merge	MS	$119
PANEL Plus	MS	$395
Pfinish - by Phoenix	MS	$239
Report Option - for Xtrieve	MS	$109
Screen Sculptor	PC	$ 89
SPSS/PC Plus	PC	$749
Synergy - create user interfaces	MS	$329
XQL - SQL for Btrieve	MS	$649
Xtrieve - organize database	MS	$179
ZAP Communications - VT 100	PC	$ 89

OS Support

DOS Merge 286	PC	$139
Microsoft Windows	PC	$ 69
Development Kit	PC	$319
MKS AWK	MS	$ 65
MKS Toolkit - Unix vi, awk	PC	$114
Norton Utilities Advanced	MS	$ 99
System V/AT Combination	PC	$489
Xtree - Professional	MS	$109

Other Languages

ACTOR	PC	$ 409
Ada Dev's Toolkit-Vol. 1 & 2	PC	$ 889
Alsys Ada w/ 4 M RAM	PC	$2995
APL*PLUS/PC	PC	$ 439
CCS Mumps - Multiuser	PC	$ 359
Microsoft MASM	MS	$ 105
Modula-2 - V3.0 Dev. System	PC	$ 219
Smalltalk/V	MS	$ 79
SNOBOL4+ - great for strings	MS	$ 80
RPG II Compiler by Lattice	PC	$ 639

RECENT DISCOVERY

Halo '88 - Update includes scanner control/scanned image manipulation, 144 devices supported, PS/2. Virtual Raster Interface for EMS. New Languages Turbo C, BASIC, MS QuickBASIC. New documentation. PC Call

Other Products

ASMLIB - 170+ routines	PC	$125
Back-It by Gazelle	MS	$119
Baler	PC	$439
Dan Bricklin's Demo II	PC	$179
Disk Technician-smart upkeep	PC	$ 89
Fast Back Plus	PC	$149
Flash-Up	PC	$ 69
Interactive Easy Flow V5.0	PC	$125
Link & Locate - Intel tools	MS	$325
Mace Utilities	MS	$ 85
MKS Trilogy	MS	$ 99
PC-Metric	MS	$ 89
PC/Tools-by Custom Software	PC	$ 69
Plink 86 Plus - overlays	MS	$289
PVCS - by Polytron	PC	$129
R-DOC/X	MS	$135
risC by IMSI - H.A.L.	MS	$ 75
Show Partner F/X	PC	$328
TLIB	PC	$ 79
Visible Computer: 8088	PC	$ 65
WKS Library by Tenon	PC	$ 79

Pascal and Supporting

ALICE - learn Pascal	PC	$ 49
Marshall Pascal	MS	$ 155
MetaWINDOWS-graphics tools	PC	$ 159
MetaWINDOWS PLUS	PC	$ 219
Microsoft PASCAL - faster	MS	$ 199
Pascal 2 - by Oregon Software	MS	$ 239
Turbo Extender by Turbo Power	PC	$ 65
TurboHALO	PC	$ 79
Turbo Optimizer - object only	PC	$ 59
Turbo Pascal - V4.0	MS	$ 65

Translator

dB2C - requires toolkit	MS	$249
RTC PLUS by Cobalt Blue	MS	$379
SofTRAN - Translation Lang.	PC	$349
TP2C	PC	$199
Turbo-to-C-Tools by TGL	PC	$479

Xenix Unix

C-Terp by Gimpel Software	$379
Cobol - by Microsoft	$649
dBQuery - multi source	Call
Fortran or Pascal-by Microsoft	$459
FoxBASE+	Call
Panel Plus	$535
RM/Cobol	$959
Xenix Complete System 286	$999
Xenix Development System 286	$499

FEATURE

Foxbase PLUS by Fox Software. DBMS conforms to standard MAC interface. Up to 9 definable output windows, definable buttons, import/export hypercard stacks. MAC. Developer Kit. $339

Call for a catalog, literature, advice, and service you can trust

LIFEBOAT INC.

678-39, Yuksam-Dong, Kamgnam-Ku, Seoul, Korea

PHONE : (02)555-1392

FAX : (02)555-9250

만능보호장치 HASP-Ⅱ

소프트웨어 보호에 대하여

상품가치가 있는 소프트웨어의 개발에는 장기간의 시일과 노력, 그리고 엄청난 비용의 투자가 수반된다. 이런면을 생각할 때 불법적인 무단복제 사용에 의하여 소프트웨어의 상품가치가 손상되는 것은 실로 안타까운 일이다. 최근 국내에서도 소프트웨어보호법에 의하여 소프트웨어저작권자의 권리를 보호하고는 있으나 법적인 보호범위와 현실적 효과에는 한계가 있으며 무단복제 사용에 대한 일차적인 대응방안을 소프트웨어제작자 스스로가 마련해야 하는 것이 현실이다.

외국의 경우도 상황은 비슷하여 보호기법과 이에 대한 복제기법이 마치 경쟁이라도 하듯 개발되는 악순환을 거듭하여 왔으며 이들 중 일부가 국내에도 도입되어 사용되고 있다. 특히, 플로피디스크를 이용하는 전통적인 보호기법은 소프트웨어적인 방법에 의존하고 있으며 그 해석 역시 동일방법으로 접근이 가능한 것으로서 이미 그 의미를 상실한지 오래다. 이와 같은 배경하에서 최근 소프트웨어적인 방법과 하드웨어적인 방법을 복합적으로 활용하는 새로운 형태의 소프트웨어 보호시스템이 등장하고 있으며 소프트웨어의 복제는 자유롭게 허용하되 소프트웨어를 작동시키기 위해서는 정교한 특수 하드웨어 장치를 컴퓨터에 접속시키도록 하는 등의 고도한 보호기능을 제공함으로써 소프트웨어 보호시스템의 새로운 장을 열고 있다.

이런 종류의 보호 시스템중 HASP-Ⅱ가 그 대표적인 예로 이를 소개함으로써 복제, 보호시스템들의 경향과 아울러 보호원리 등을 이해하는데 조금이나마 도움이 됐으면 하는게 본 편집부의 조그만 바램이다.

소프트웨어 보호시스템의 종류

소프트웨어 보호시스템은 여러 종류가 있으나 유형별로 분류하면 다음과 같다.

(1) 응용프로그램논리에 의한 보호

초기의 원시적인 형태로 응용프로그램내에서 프로그램작동에 관련되는 조건을 비교하여 일정조건하에서만 프로그램을 작동시키는 방법으로써 작업횟수의 카운트나 작업일자 등을 미리 수록한 값과 비교하는 것 등이 이에 속한다. 가장 손쉬운 방법이기는 하나 운영이 불편하며 쉽게 해석이 된다는 단점이 있다. 또한 상호명 등을 응용프로그램의 곳곳에서 출력시키는 방법도 있으며 기계어코드의 분석을 통하여 제거하지 못하도록 아스키코드값을 변형시키는 암호화(Encryption)기법이 쓰이기도 한다.

(2) 런모듈(Run Module)을 변형처리하는 방법

가장 일반적인 형태의 보호기법으로써 플로피디스크에 수록된 런모듈을 변형처리하여 카피명령이나 디스크카피 등 도스의 표준적인 방법으로는 복제할 수 없는 형태로 가공하는 기법이며 이는 소프트웨어 보호비용이 비교적 저렴한 반면 소프트웨어적인 접근방법에 의해 쉽게 해석될 수 있으며 실제 대부분의 보호기법이 이미 해석되어 있다.

그리고 이 방법은 복제를 금지시키는 방법으로서 사용자가 백업을 자유롭게 할 수 없고 소프트웨어를 하드디스크에 설치, 사용하는데 제약이 따른다. 또한 보호처리된 런모듈 화일이 손상되기 쉽다는 점과 알엠코볼(RM/COBOL)과 같이 실행화일(EXE 혹은 COM화일)을 생성하지 않는 언어에는 적용이 불가능하다는 점등이 최대 취약점이라고 볼 수 있다. 게다가 복제전용기기가 아닌 일반 퍼스널컴퓨터를 이용해서 보호처리된 원본과 동일하게 사본을 얻을 수 있는 카피옵션보드도 개발되어 있는 실정이다.

(3) 하드웨어적인 방법

컴퓨터본체에 암호가 수록된 인쇄회로기판을 내장시켜 응용프로그램에서 그 암호값을 확인케 하는 하드웨어키로서 소프트웨어의 사용이 인쇄회로기판을 내장시킨 특정컴퓨터에만 국한된다. 그러나 이 경우도 응용프로그램의 런모듈을 분석하여 보호기능을 제거시킬 수 있으며 회로판의 복제생산이 가능하다. 그리고 사용자가 이 방법을 적용하고자 할 경우 인쇄회로기판(P.C.B)을 설계, 제작해야 한다는 부담이 따르기 때문에 이또한 훌륭한 보호시스템으로 보기에는 어렵다.

(4) 하드웨어와 소프트웨어의 복합적인 방법

위에서 언급한 보호시스템은 모두 보호기능이 불완전하여 소프트웨어사용자에게 불편을 초래하고 있다. 최근 복제기술이 발달하여 플로피디스크에 의한 보호방법이 유명무실해짐에 따라 소프트웨어보호의 새로운 방법이 필요하게 되었다. 이 방법은 정교한 하드웨어장치와 보호기능을 제공해주는 소프트웨어의 결합에 의하여 기존 보호시스템의 불완전함과 단점을 극복하고 있으며 필자의 견해로는 이 방법이 당분간은 안전한 보호시스템으로서의 자리를 지킬 수 있으리라 추측된다.

참고로 위에서 소개한 각 분류별 비교를 표 1에 나타내었다.

HASP-Ⅱ 보호시스템

HASP는 Hardware for Software Protection의 약자로서 하드웨어와 소프트웨어를 결합한 보호장치의 대표적인 예이다.

이 시스템은 1987년 이스라엘의 ALADDIN KNOWLEDGE SYSTEM사에서 개발된 것으로 다른 보호장치와는 달리 보호효과가 크며 사용이 간편하고 기존의 하드웨어와 소프트웨어에 영향을 주지 않는 등 종래의 보호장치의 단점과 한계를 극복한 새로운 형태의 보호장치로서 작은 성냥크기의 플라스틱 케이스에 내장된 전자회로 및 관련 소프트웨어로 구성된다.

〔표 1〕 보호장치의 특징 비교

	HASP-Ⅱ	보호처리된 디스크	P.C.B 보드
방식	소프트웨어+하드웨어	소프트웨어	하드웨어
원리	복제 허용 하드웨어 키	복제 금지	패스워드 키
효과	효율적인 보호기능 제공	카피 유틸리티 및 카피 보드에 의해 보호기능 상실	디버그에 의한 보호기능 제거 가능
백업	자유	부자유	자유
접속/분리	컴퓨터 외부에서 패러럴포트에 접속		PCB를 슬롯에 삽입하기 때문에 S/W 사용이 특정 컴퓨터로 제한됨
하드디스크	가능	부자유	가능
기타	신제품, 수입품		슬롯 점유

(1) HASP-Ⅱ 플러그

HASP-II 플러그 속에는 고유한 알고리즘을 형성하는 전자회로가 들어있는데 HASP-II에 의해 보호처리된 소프트웨어만이 이 알고리즘을 인식할 수 있다. 즉 보호처리된 소프트웨어는 실행하기에 앞서 플러그의 알고리즘을 확인해야만 실행되는 것이다.

보호처리에 앞서 컴퓨터의 뒤쪽을 보면 프린터와 연결되는 패러럴포트가 있는데 여기에 플러그를 접속하면 준비는 끝난 셈이다. 프린터를 사용하는 경우는 플러그의 뒷쪽으로 프린터의 케이블을 연결하면 된다. HASP-II 플러그 속에 있는 전자회로에는 프린트제어코드를 인식할 수 있는 부분이 있어서 프린터에 해당되는 코드들은 자기와 무관함을 알고 뒤로 흘려보내기 때문에 프린터의 동작에는 영향을 끼치지 않는다(그림 1 참고).

화면 1은 임의의 인수값을 주었을 때 플러그에서 수치를 만드는 몇가지 예를 보인 것인데 이것은 플러그에서 출력되는 값을 여섯개의 레지스터에 기억시켰다가 화면에 출력하도록 간이 프로그램을 작성하여 실행시킨 결과이다.

(2) HASP-Ⅱ 유틸리티 및 서브프로그램

HASP-II 플러그와 함께 제공되는 디스크(1장)에는 각종 환경에서 이용할 수 있는 화일들이 들어 있어서 매크로어셈블러, 시(C), 파스칼, 엠에스코볼, 베이직, 디베이스Ⅲ+ 등 대부분의 언어환경을 지원할 수 있다.

주요 구성화일을 살펴보면 다음과 같다.

- **READ.ME** : HASP-Ⅱ시스템의 개요를 담고 있다.
- **HASPINST.EXE** : 실행화일을 보호처리하는 메인 인스톨화일이다.
- **HASPINST.TXT** : HASPINST.EXE와 함께 있어야 하며, 보호처리시 필요한 메시지를 담고 있다.
- **HASPRES.EXE** : 선택사항 이용을 위한 메모리 상주 프로그램이다.

다음은 각종 언어환경을 대비해 놓은 서브디렉토리 이름들로서 오른쪽에는 해당 디렉토리내에서 이용 가능한 언어를 나타낸 것이다.

- **MSLINK** : 마이크로소프트 베이직, 파스칼, 코볼, 어셈블러
- **MSCLINK** : 마이크로소프트 시(C)
- **TURBO** : 터보파스칼(버전 3.0과 4.0), 베이직, 시(C)
- **MT86** : MT86 파스칼, 어셈블러
- **RMCOBOL** : 알엠코볼
- **LATTICE** : 래티스 시(C)
- **CLIPPER** : 클리퍼 디베이스Ⅲ+ 컴파일러
- **REALCOB** : 리얼리어코볼(Realia Cobol)
- **RM85COB** : RM85 코볼

(3) HASP-Ⅱ를 이용한 화일 보호

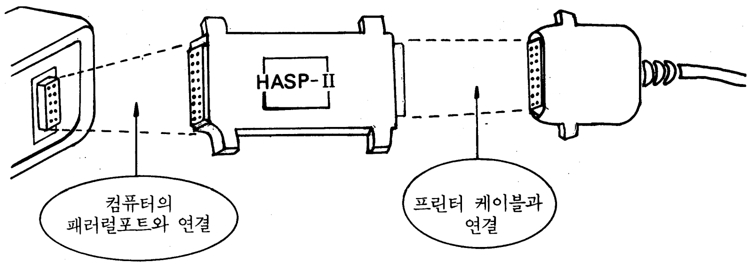
앞서 설명했듯이 HASP-II는 소프트웨어와 하드웨어의 결합에 의한 보호원리를 가지고 있다.

보호대상 화일이 즉시 실행가능한 목적화일이면 다음과 같이 입력한다.

```
HASPINST [드라이브 명] <화일 사양> ↵
```

그러면 HASPINST.EXE가 로드되면서 사용자가 지정하는 메시지와 함께 플러그속의 전자회로(알고리즘)를 인식하는 루틴을 해당화일에 첨가하고 프로그램코드를 암호화하여 다시 세이브한

〔그림 1〕 HASP-Ⅱ 플러그 접속방법



다.

화면 2는 사용자들에게 익히 알려진 PCTOOLS.EXE을 보기로 보호처리과정을 보인 것이다. 그리고 여기서 알고리즘을 추적하기 위해 첨가되는 루틴은 매우 복잡한 코드로 이루어져 있기 때문에 보호처리 되기전과 후의 화일크기를 비교해보면 10K바이트 정도 차이가 있음을 볼 수 있다(화면 3 참조).

이렇게 해서 새로 구성된 화일은 실행되기 전에 HASP-II 플러그를 체크한 후 알고리즘을 거쳐 되돌려진 값을 조사하게 된다. 만일 플러그가 패러럴 포트에 없거나 다른 종류의 플러그가 접속되어 틀린 수치가 입력되면 사용자가 지정해 준 에러메시지를 출력하고 실행을 중단시켜 버린다. 위의 경우를 예로 든다면

A>PCTOOLS ↵

를 입력하여 실행시켜 보면 아직 HASP-II 플러그가 컴퓨터 뒤쪽의 패러럴 포트에 접속되어 있는 상태이기 때문에 순조롭게 실행된다. 그러나 HASP-II 플러그를 빼고 다시 실행시키면 이번엔 다음과 같이 위의 인스톨과정에서 사용자가 지정한 메시지가 출력되고 실행이 중지된다.

*** I'm Sorry But You Have Not HASP-II ***

이번엔 종류가 다른 플러그를 접속하고 위 과정을 수행하면 또한 이 경우를 대비해서 사용자가 지정한 메시지가 출력되고 실행이 중지된다.

*** This Plug Is Not Correct ***

그리고 베이직이나 알엠코볼과 같이 실행화일을 만들 수 없는 경우는 HASP-II의 서브프로그램을 호출(CALL)하는 루틴을 해당화일과 연결해서 이용

〔화면 1〕 임의의 값을 주었을 때 플러그에서 나오는 수치

```
Enter Input Key Value : 12

Return Value 1    2626  -14996
Return Value 2    2626  -14996
Return Value 3    2626  -14996

Press Enter Key to Continue ...

Enter Input Key Value : 10

Return Value 1   22594  -15026
Return Value 2   22594  -15026
Return Value 3   22594  -15026

Press Enter Key to Continue ...

Enter Input Key Value : 00

Return Value 1   22082  -14920
Return Value 2   22082  -14920
Return Value 3   22082  -14920

Press Enter Key to Continue ...
```

〔화면 2〕 보호처리중에 나타나는 화면 메시지

```
A:\>HASPINST PCTOOLS.EXE
Hasp-II (tm) V1.6a
Hardware for Software Protection
Copyright (c) 1987 , 1988 - Aladdin Knowledge System Ltd.

Make sure your HASP-II plug is connected and hit <ENTER>

If you have another HASP-II plug with a different code, it is
possible to use more than one code to protect this program.
Do you want another HASP-II code to open this program (y/n) ? n

Message for plug not found is :
HASP-II plug not found.
Press <ENTER> to leave as is
or type your own preferred message and press <ENTER>:*** I'm Sorry But You
Have Not HASP-II ***

Message for illegal plug is :
Illegal HASP-II plug.
Press <ENTER> to leave as is
or type your own preferred message and press <ENTER>:*** This Plug Is Not
Correct ***

Hasp-II done
```

필요에 따라 수개의 플러그를 하나의 화일에 할당할 수 있다.

플러그가 없을 때의 메시지를 지정한다.

다른 종류의 플러그가 접속되어 있을 때의 메시지를 지정한다.

〔화면 3〕 보호처리 전과 후의 디렉토리

```
A:\>DIR

 Volume in drive A has no label
 Directory of A:\

 PCTOOLS  EXE     83840   9-14-85    1:19p
 HASPINST EXE     57672   5-30-88   11:19p
 HASPINST TXT       862   4-18-88    2:47p
         3 File(s)  219174 bytes free
```

(a) 보호처리 전

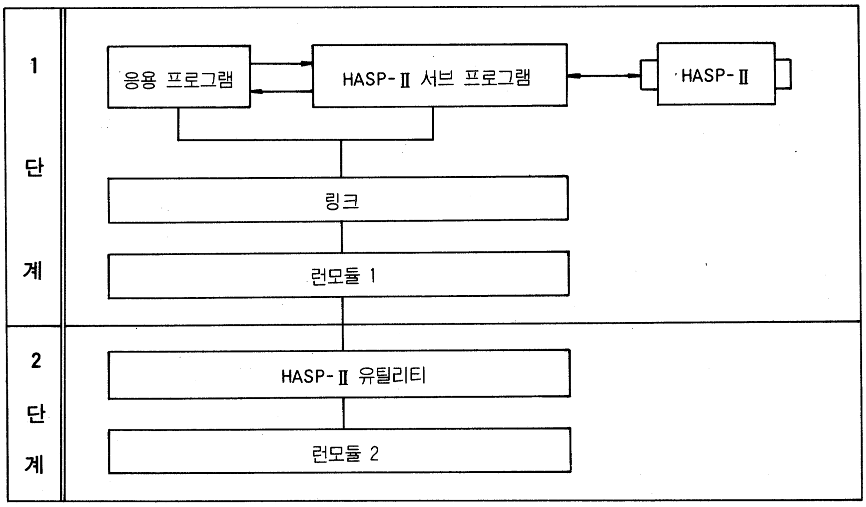
```
A:\>DIR

 Volume in drive A has no label
 Directory of A:\

 PCTOOLS  EXE     93840   9-14-85    1:19p
 HASPINST EXE     57672   5-30-88   11:19p
 HASPINST TXT       862   4-18-88    2:47p
         3 File(s)  209174 bytes free
```

(b) 보호처리 후

〔그림 2〕 보호절차의 단계별 분류



하면 된다. 예를 들어 알엠코볼인 경우, DEMO.CBL이란 소스화일에 서브프로그램 HASPRM.EXE를 호출하는 루틴을 첨가하고

```
A>RMCOBOL DEMO ↵
```

를 입력하여 실행시키면 DEMO.COB을 얻게 되는데 이 화일은 이미 보호처리가 되어 있는 상태가 된다. HASP-II가 접속되어 있는 상태에서

```
A>RUNCOBOL DEMO L=HASPRM.EXE ↵
```

를 입력하면 제대로 실행되고 플러그가 없으면 에러메시지가 출력되고 실행을 멈추고 만다. 그외 내용은 앞에서 설명된 것과 같다.

참고로 소스화일과 실행화일이 둘 다 존재할 때는 먼저 소스화일을 보호처리하고(1단계) 이것을 컴파일하여 얻은 목적화일을 다시 보호처리함으로써(2단계) 더욱 강력한 보호효과를 얻을 수 있다(그림 2 참고). PC

디버깅

● 정정합니다

—'88년 12월호 230쪽 도로스터 TURBO XT의 가격은 82만원이 아니라 130만원이기에 바로 잡습니다.

—'88년 12월호 237쪽 Dragon 386UX의 가격은 660만원이 아니라 990만원이기에 바로 잡습니다.

●제작코너●

31개 음의 표현이 가능한 음성출력시계

정일준/한양대 전자과

컴퓨터를 가지고 무엇인가를 한다는 것은 상상만큼이나 재미있고 또 흥미로운 일이다. 더우기 그 내부구조를 파헤쳐가면서 점점 컴퓨터를 더 알게 되고 그래서 많은 부분에 이용할 수 있게 된다면 그 흥미는 훨씬 커질 것이다. 컴퓨터는 그 자체로 만능기계는 결코 아니다. 하지만 이용만 잘한다면 그와 유사한 정도까지 사용할 수도 있다.

이번 제작은 하드웨어적으로 컴퓨터를 전혀 사용하지 못하고 있는 많은 독자들을 위해 시도되었다. 컴퓨터를 이해하기 위해서는 이론적인 면과 실제적인 면, 그리고 하드웨어적인 면과 소프트웨어적인 면에 대해 모두 어느 정도 이상의 지식과 경험이 필요한데 이번 제작은 이 모든 부분에 대해 최소한의 지식과 경험은 제공할 수 있으리라 생각된다. 따라서 이번 기사를 쓰면서 필자는 이런 점에 유의하였다.

하드웨어적으로는 가장 간단한 전원부분에서 부터 시작하였고, 그리고 만능소자라고도 불리워지는 연산증폭기(OP Amp)를 비교적 자세히 설명하였다. 하나하나 실험해보기 바란다. 소프트웨어적인 면은 동작설명이 조금 소홀한 느낌이 없지 않으나 이것은 여러 어셈블러에 대한 서적을 참고하여 보충하기 바란다. 하지만 실제 실행하는 방법등에 대해선 많은 지면을 할애하여 설명하도록 노력하였다.

제작

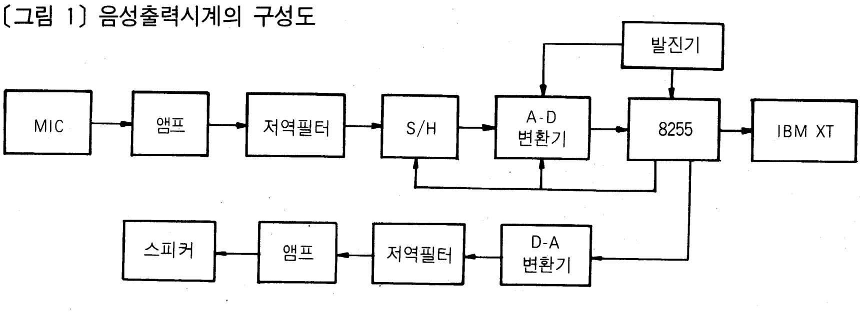
음성출력시계의 구성도는 그림 1과 같다.

마이크를 통해 입력된 신호는 우선 앰프에 의하여 크게 증폭된 후 저역필터(Low Pass Filter)로 들어간다. 저역필터는 지정된 주파수 이하의 신호만 통과시키는 역할을 하는 회로이다. 따라서 필요없는 고주파를 제거하는 작용을 한다. 고주파가 제거된 신호는 샘플홀드 회로에 의하여 A-D변환이 끝날 때까지 일정한 값으로 유지된다. 만약 A-D변환을 하는 도중에 입력신호가 변한다면 올바로 변환이 진행되지 않기 때문이다. 이렇게 A-D변환이 된 신호는 8255병렬포트에 의하여 컴퓨터의 메모리에 저장되게 된다.

음성출력의 경우는 거의 음성입력 방법의 역순이다. 즉, 프로그램이 실행되면 시스템 콜을 사용하여 시각을 알아내고 이에 해당하는 메모리상의 데이터를 8255병렬포트를 통해 내보낸다. 이 데이터는 D-A 변환기를 통과하면서 아날로그 신호로 바뀌게 되고 저역필터에 의해 고주파잡음이 제거된 후 크게 증폭되어 스피커를 통해 출력되는 것이다. 모든 타이밍은 발진기가 제공한다. (본 제작에서는 6KHz)

전체적인 흐름을 대략 살펴보았는데 하드웨어를 처음 시작하려는 사람의 경우는 조금 난해한 부분도 없지 않을 것이다. 따라서 제작순서는 되도록 쉬운 것부터 시작하기로 하겠다. 또 하나, 이 제작을 만들기 위해선 꼭 갖추어야 할 몇가지 도구가 있다. 우선은 납과 납땜인두가 필요하다. 납은 가는 것이 보다 편리하고 납땜인두는 20W가량의 것이며 무난하리라 본다. 납땜실력이 결국 성공이냐 실패냐를 가름짓는 가장 큰 요인이 되므로 충분히 연습을 하여야 할 것이다. 그 다음으로는 테스터가 있어야 한다. 테스터가 없이 작품제작에 성공한다면 오히려 이상하게 느껴질 만큼 테스터는 없어서는 안 될 도구이다. 또한, 여기에 절대적으로 필요한 것은 아니지만「Bread board」라고 불리는 실험용 기판이 있으면 아주 좋다. 이것은 납땜을 하지 않고도 회로의 작동여부를 알아볼 수 있어 대단히 편리하게 되어있다. 이 밖에 니퍼, 드라이버, 핸드드릴 정도는 갖춰 놓아야 할 것이다.

〔그림 1〕 음성출력시계의 구성도



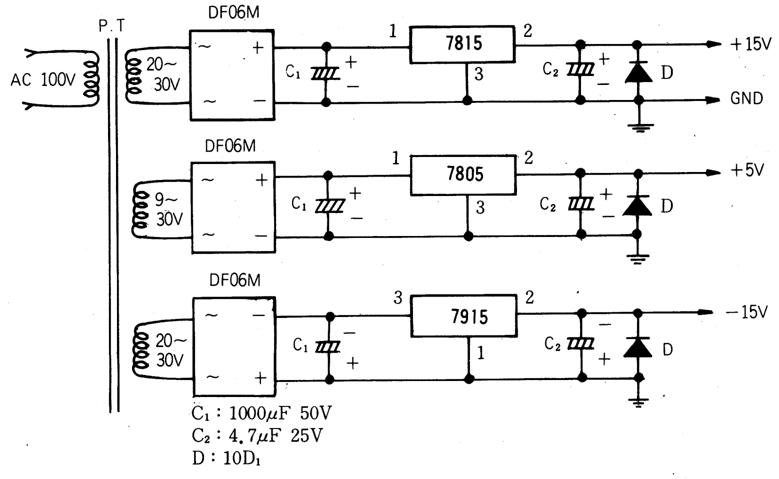
전원부의 제작

작품제작에 있어 가장 먼저 준비되어야 할 부분은 역시 전원부이다. 더우기 ±15V의 전압이 필요한 본 작품의 경우는 적당한 전원을 구하기가 어렵다. 따라서 정전압 IC를 사용하여 간단히 만들어 보기로 하자. 회로도를 그림 2에 보였다.

여기에 사용된 특별한 부품이라면 μA7800시리즈와 μA7900시리즈 정도일 것이다. μA7800시리즈는 플러스전압의 3단자전압안정기이고 μA7900시리즈는 마이너스전압의 3단자전압안정기이다. 이 두 IC의 핀배치를 **그림 3**에 보였다. 두가지 모두 최대 출력전류가 1A이고 최대입력전압은 약 35V이다. 형번호를 보면 출력전압을 알 수 있는데 μA7805는 플러스 5V 출력이고 μA7915는 마이너스 15V 출력을 나타낸다. 보다 자세한 것은 규격표를 참고하기 바란다.

파워트랜스는 2A이상의 것을 쓰기 바란다. 최악의 경우 3A이상을 견뎌내야 하지만 본 작품의 경우는 그럴 필요까지는 없다. 그리고 IC에는 반드시 방열판을 붙이기 바란다. 되도록이면 각각 하나의 방열판에 따로 붙여 쇼트되지 않게 조금 떨어뜨려 놓으면 좋다. 만약 부득이 하나의 방열판에 모두 붙여야 할 경우는 마이카와 나이롱 비스를 사용하여 반드시 절연시키길 바란다.

〔그림 2〕 전원부



발진기의 제작

발진기는 A−D 변환기 구동용의 600kHz 와 샘플링 속도를 제공하기 위한 6kHz를 모두 발생시켜야 한다. 나중에 자세히 설명하겠지만 A−D 변환기 구동에는 10~1280kHz가 필요하고 샘플링속도는 사용자의 필요에 따라 적당히 조정할 수 있다. 물론 IBM−PC 상에서 4.77MHz의 클럭을 출력하여 사용할 수도 있다. 이 때는 카운터부만을 만들면 된다. 회로도를 **그림 4**에 보인다.

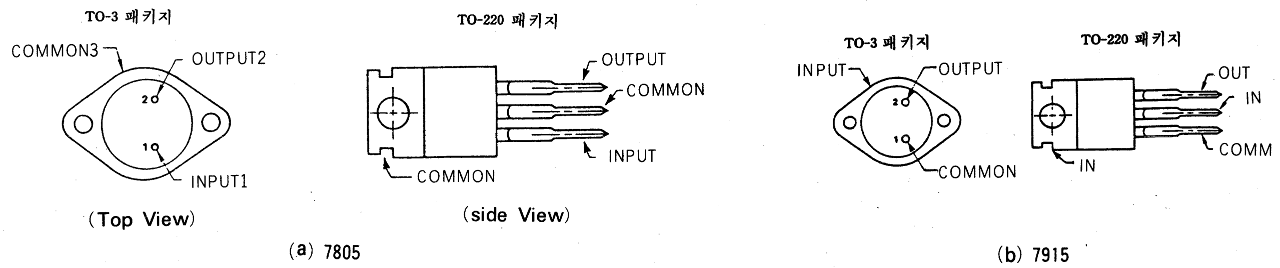
74LS04와 74LS390의 핀접속도를 **그림 5**에 보인다. V_{CC}에는 +5V를 접속하고 사용하지 않는 모든 입력핀은 V_{CC}에 접속하기 바란다. 74LS390은 입력A에 출력 Q_D를 접속하여야 2−5진 카운터로 동작한다는 사실을 명심하여야 할 것이다.

완성이 되었으면 동작여부를 테스트 해보아야 한다. 오실로스코프가 있어야 그 파형을 정확히 볼 수가 있는데 아마 오실로스코프를 가지고 있는 독자는 거의 없을 것이다. 조금 불안하지만 테스터로 전압을 재어 ϕ_1과 ϕ_2의 전압이 대략 2.5V가 나오면 일단 동작이 되는 것으로 믿고 다음 제작으로 넘어가도록 한다. ϕ_2의 경우는 6kHz이므로 귀로 들어 확인할 수도 있지만 이렇게 하려면 버퍼와 저주파 전력증폭기가 있어야 한다. ϕ_2에 바로 스피커를 접속하면 IC가 파괴되는 경우가 있으므로 주의하기 바란다.

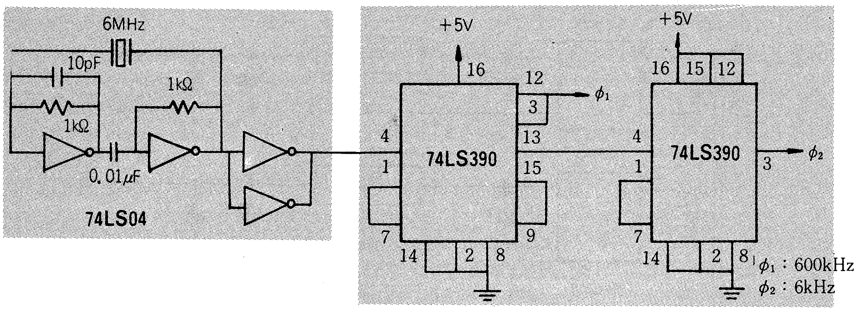
8255 병렬포트의 제작

8255A는 주로 80계열에 사용되는 범용의 프로그래머블 I/O 인터페이스이다. 패러럴포트 중에서도 가장 많이 사용되는데 그 이유는 무엇보다도 입출력의 핀수가 많기 때문이다. PA~PC까지 모두 24핀을 사용할 수 있으므로 웬만한 것은 모두 접속할 수 있다. 그

〔그림 3〕 7800과 7900의 핀배치도

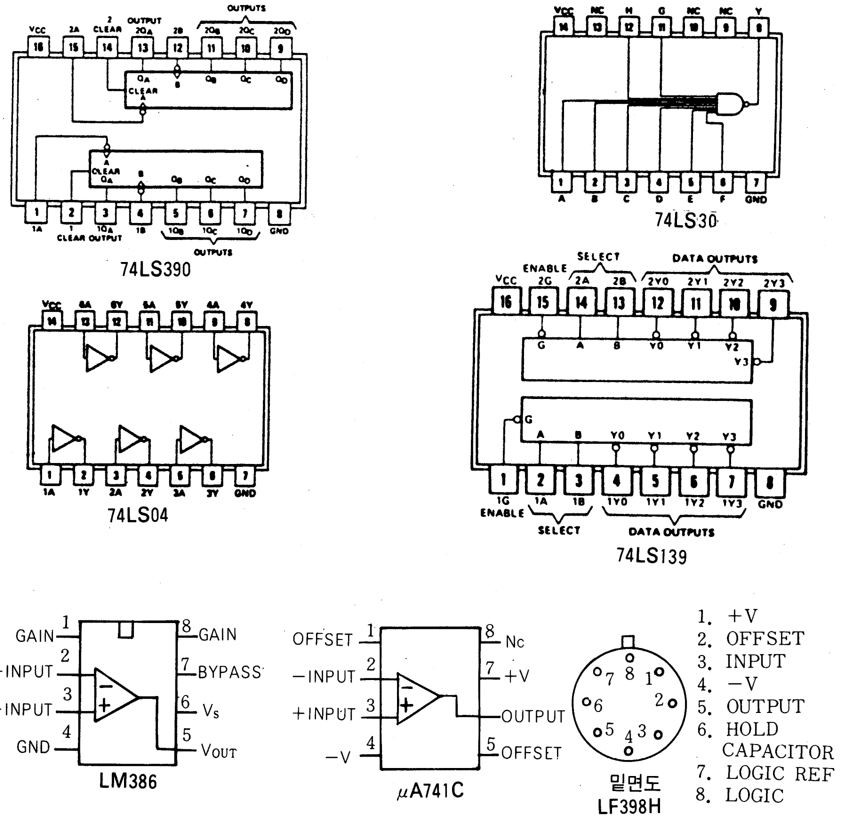


〔그림 4〕 발진기



(a) 발진부 (b) 카운터부

〔그림 5〕 여러 소자의 핀배치도



림 6에 블럭도와 핀 접속도를 보인다.

핀 D_0~D_7은 CPU의 D_0~D_7에 연결하면 되고 $\overline{WR}$은 $\overline{IOW}$에 $\overline{RD}$는 $\overline{IOR}$에 연결한다. 자세한 것은 **그림 7**의 회로도를 참고하기 바란다.

8255A에는 모드 0, 모드 1, 그리고 모드 2의 3가지 모드가 있다. 여기서 모드 0은 가장 기본적인 I/O포트이고, 모드 1은 8비트 단위의 핸드세이크 모드, 모드 2는 쌍방향 핸드세이크 모드이다. 자세한 것은 여러 참고도서를 참조하기 바라고 여기서는 이번 제작에 필요한 모드 0에 대해서만 간단히 설명하겠다.

우선 **그림 8**를 보자.

모드 0으로 동작하기 위해서는 기본적으로 D_7에는"1", D_6와 D_5는 모두"0"으로 세트되어 있어야 한다. (D_2도 물론"0"으로 세트). 이 상태에서 D_4는 PA의 입출력을, D_1은 PB의 입출력을, 그리고 D_0와 D_3의 입출력을 관장한다. 만약 전포트를 입력으로 하려면 모두 1로 세트하면 되고 포트 A만을 입력으로 하려면 D_4만을 1로 하면 된다.

이제 IBM−PC XT의 I/O 채널을 살펴보자. (**그림 9 참조**)

여기서는 이번 제작에 관계된 신호만을 간단히 설명하기로 하겠다.

RESET DRV는 시스템 논리회로의 전원을 켜거나 저전압을 걸어서 리세트 또는 초기화할 때 사용한다. 이 신호는 클럭의 하강 모서리에서 발생하며 하이레벨에서 활성화된다.

A_0~A_{19}는 시스템에서 메모리와 I/O 기기에 어드레스를 지정할 때 사용된다.20개의 어드레스선은 1M바이트 이상의 메모리를 호출할 수 있다. A_0는 최하위 비트이고 A_{19}는 최상위 비트이다. 본 작품에서는 8255A의 $\overline{CS}$단자를 제어하기 위해 사용된다.

D_0~D_7은 프로세서, 메모리, I/O 소자의 데이터버스 비트 0 ~7 에 연결되며, D_0는 최하위 비트, D_7이 최상위 비트이고 하이레벨에서 동작한다.

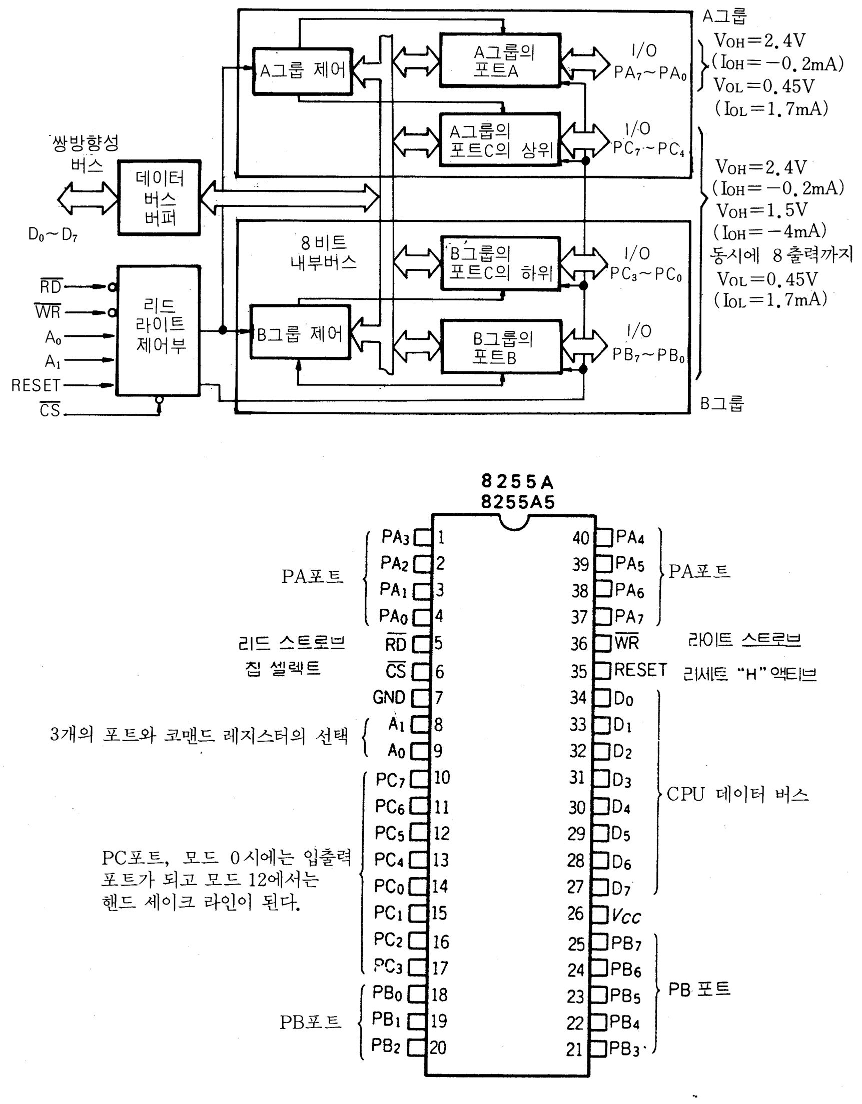
ALE(어드레스 래치 인에이블)은 8288버스 콘트롤러에 의해 제공되며 프로세서로부터의 유효한 번지에 래치하기 위해 시스템 보드상에서 이용된다.

$\overline{IOR}$은 I/O 소자가 데이터 버스를 통해 데이터를 읽어들이도록 한다. 프로세서나 DMA 콘트롤러에 의해 활성화되며"L"의 상태에서 동작한다.

$\overline{IOW}$는 I/O 소자가 데이터 버스를 통해 데이터를 내보내도록 한다. 프로세서와 DMA 콘트롤러에 의해 활성화된다.

AEN은 I/O 채널에서 DMA 전송이 이루어지도록 다른 소자를 디게이트하는데 사용한다. "H"상태에서 어드레스 버스, 데이터 버스, 읽기 명령라인, 쓰기 명령라인이 DMA 콘트롤러에 의해 제어된다. 이밖의 신호선에 대

〔그림 6〕 8255A의 블록도와 핀배치



해서는 IBM사의「IBM Technical Reference」를 참고하기 바란다.

이제 회로를 살펴보자. 우선 IBM-PC XT의 경우는 확장버스가 슬롯형식이기 때문에 슬롯이 필요하다. 필자의 경우는 슬롯 부분만을 직접 만들어 사용했는데 시중에서 슬롯 부분이 달린 만능 기판을 사서 만들어도 괜찮다. 74LS30은 8입력 NAND 게이트이므로 A_4~A_8이 1, A_9와 AEN이 0일때만 그 출력이 0이 된다. 그리고 74LS139는 라인 디코더로서 G, B, 그리고 A가 모두 0일 때만 출력이 0이 되어 8255A가 작동하게 된다. 따라서 여기에 해당하는 어드레스 신호가 주어지면 8255A를 사용할 수 있게 된다.

회로가 간단하기 때문에 제작하기는 어렵지 않으나 선이 매우 많으므로 주의하기 바란다. 쓰지 않는 IC의 입력 핀은 모두 +5V에 접속하기 바라고 전원은 모두 IBM상에서 뽑아쓰도록 해야 한다. 그리고 GND는 전원부의 GND와도 연결해 주어야 한다.

필터와 앰프의 제작

이 부분은 아날로그 신호를 처리하는 부분이기 때문에 아무래도 좀 더 주의를 기울여 제작하여야 한다. 부품수도 가장 많기 때문에 그만큼 복잡하기도 하다. 우선 회로도를 보자.(그림 10 참조)

모든 회로는 OP앰프를 중심으로 설계되어 있다. 여기서는 μA741C를 사용하였는데 LF356이나 LM318등을 사용하여도 무관하다. 단 이때는 오프셋 조정등을 변경하여야 한다. 규격표를 참고하기 바란다.

우선 A-D 콘버터로 들어가는 회로를 살펴보면 처음 3단은 마이크에서 들어오는 미약한 신호를 크게 증폭하는 역할을 한다. 1단만으로도 가능하나 그렇게 하면 아무래도 잡음이 섞일 위험이 많다. 증폭도는 그림 11의 OP앰프 기본회로에서 볼 수 있듯이 계산하면 된다. 예를 들어 처음 1단의 증폭도를 계산해보면

$$A=1+\frac{R_F}{R_1}=1+\frac{300K}{51K}=7\text{배}$$

가 됨을 알 수 있다.

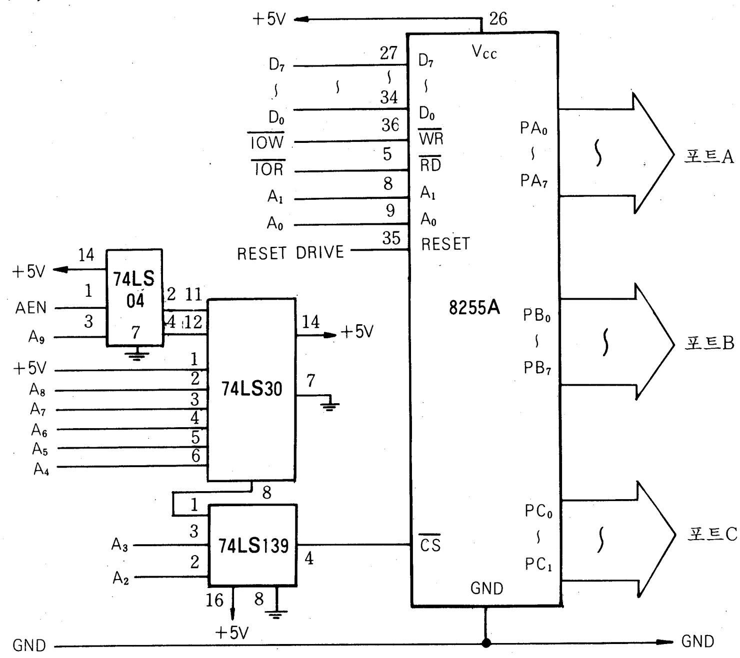
그리고 마지막단은 저역필터이다. R과 C의 위치를 바꾸면 고역이 됨을 참고로 알아두자. 이 두 필터를 적당히 조합시키면 대역필터나 밴드블로킹 필터도 간단히 만들 수 있다. 관심이 있다면 대역 필터를 적당히 조합하여 파형분석장치나 그래픽이퀄라이저 등을 만들어 보기 바란다. 저역필터의 컷오프 주파수에 따른 R과 C를 구하는 방법을 잠시 살펴보면, 만약 컷오프 주파수가 3kHz, C의 값을 0.002μF이라 가정하면

$$R=\frac{1}{2\pi f_C C}=\frac{1}{2\pi(3000)(0.002\times10^{-6})}=27\text{k}\Omega$$

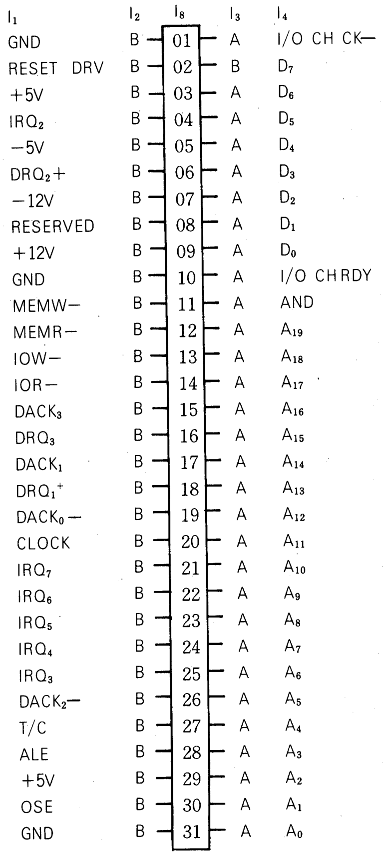
이 된다.

이제 D-A 변환기에 접속되는 부분을 살펴보면 첫째단이 저역필터이고 다음단은 저주파 전력증폭회로이다.

〔그림 7〕 8255A 병렬포트의 회로도



〔그림 9〕 I/O 채널



만약 음질이 떨어진다면 카세트의 입력단자에 연결하여 사용하는 것도 좋다.

제작해보면 알겠지만 이 필터와 앰프부분은 아무래도 여러가지 시행착오가 있기 마련이다. 따라서 처음에 기술했듯이 실험용 기판에 우선 제작을 한 후 기판으로 옮겨 납땜하는 것이 좋을 듯하다. 회로도에 패스콘을 몇군데는 표시하였으나 이것으로는 부족한 것 같고 여러가지로 실험하여 적당한 곳에 패스콘을 붙이기 바란다. 일반적으로 전원배선에 연결된 콘덴서는 10μF 이상을 달아 주어야 하고 변환기나 앰프의 출력단에는 수pF~수백pF의 콘덴서를 달아주면 좋다.

A−D 변환부의 제작

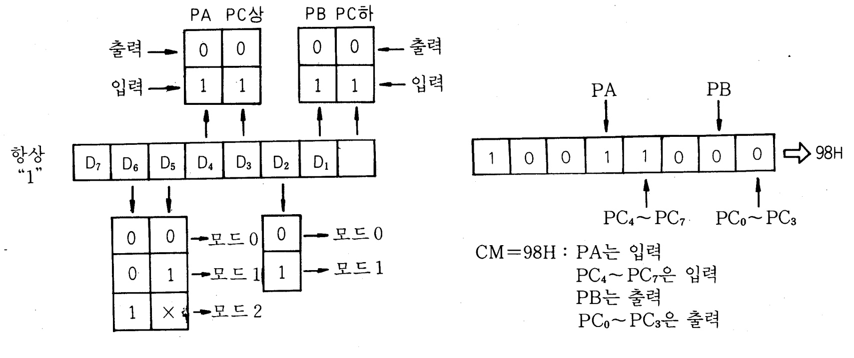
A−D 변환기는 아날로그 신호를 디지탈 신호로 바꾸는 장치이다. 자연계의 물리량, 이를테면 전압, 시간, 변위, 속도, 압력등은 아날로그량이다. 그러나 전자계산기에서는 디지탈량으로 처리하는 편이 편리하다. 그래서 아날로그량을 디지탈량으로 변환할 필요가 있는 것이다.

A−D 변환방식은 크게 계수형, 적분형, 비교형, 그리고 비직선형으로 나눌 수 있다. 여기서는 이 중 이번 제작에 사용될 비교형 특히 축차비교형 A−D 변환기에 대해서만 잠시 설명하겠다.

축차비교형의 A−D 변환기는 내부에「기준전압」으로 불리는 몇 개의 전압을 갖고 있다. 이 전압은 2진수의 각 비트에 대응하는 크기로 되어 있다. ADC0809 는 8비트이기 때문에 8개의 기준전압을 갖는다. 즉 2^0= 1, 2^1= 2, 2^2= 4, 2^3= 8, 2^4= 16, 2^5= 32,…, 2^8=256 등을 갖고 있다. 이렇게 하면 어떤 입력신호가 주어졌을 때 2^8즉 256 개로 분할하여 0~255의 수치로 표현할 수 있는 것이다.

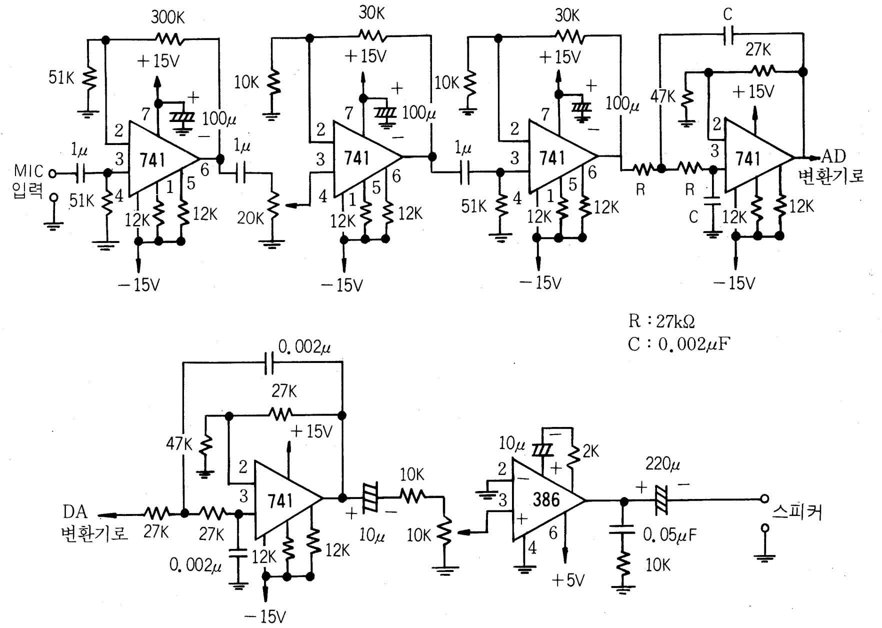
실제로 ADC0809는 0~5V의 입력

〔그림 8〕



(a) 코맨드 레지스터의 기입형식　(b) 모드 0 의 코맨트 레지스터의 기입 예

〔그림 10〕 필터와 앰프



신호를 가지기 때문에 약 0.02V의 변화까지도 알 수 있게 된다. 하지만 본 제작에서는 단지 4비트의 변화만을 이용하게 된다. 메모리의 양을 줄이기 위하여 이렇게 한 것인데 사용해본 결과 전혀 문제가 없었다.

ADC0809의 여러 데이터를 **그림 12**에 보였다. 따로 응용할 때 참고하기 바란다.

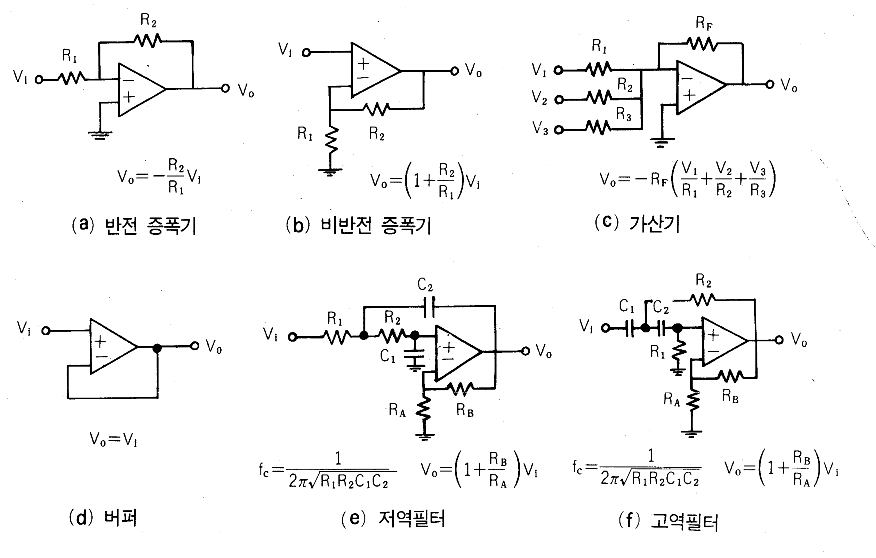
우선 A－D변환부의 회로도를 살펴보자.(그림 13 참조)

배선은 매우 간단하기 때문에 쉽게 제작할 수 있을 것이다. 바이패스(패스콘)을 잊지 말고 붙이자.

첫단은 샘플홀드 회로이다. OP앰프를 써서 만들수도 있으나 샘플홀드 IC인 LF398H를 이용하는 것이 간단하고 실패율도 적다. C_H는 홀드 콘덴서로 입력신호를 일정시간동안 유지하는 역할을 한다. C_H가 크면 정밀도는 좋아지지만 애퀴지션 시간이 길어져 속도가 떨어지고 C_H가 작으면 속도는 빠르지만 아무래도 정밀도가 떨어진다. 필요에 따라 적당한 값의 콘덴서를 쓰기 바란다. 또한 C_H는 되도록 필림콘덴서를 써야 한다. 부득이할 경우는 마일라를 써도 괜찮다.

둘째단은 입력신호의 전압을 2.5V 만큼 높여주는 역할을 한다. 10kΩ의 가변저항을 적당히 조정하여, 입력신호를 GND시켰을 때 ADC0809의 26번 핀의 전압을 2.5V로 맞추면 된다. 이렇게 하는 이유는 ADC0809의 입력신호범위가 0~5V이기 때문이다. OP앰프 자체는 버퍼로서 사용된다.

〔그림 11〕

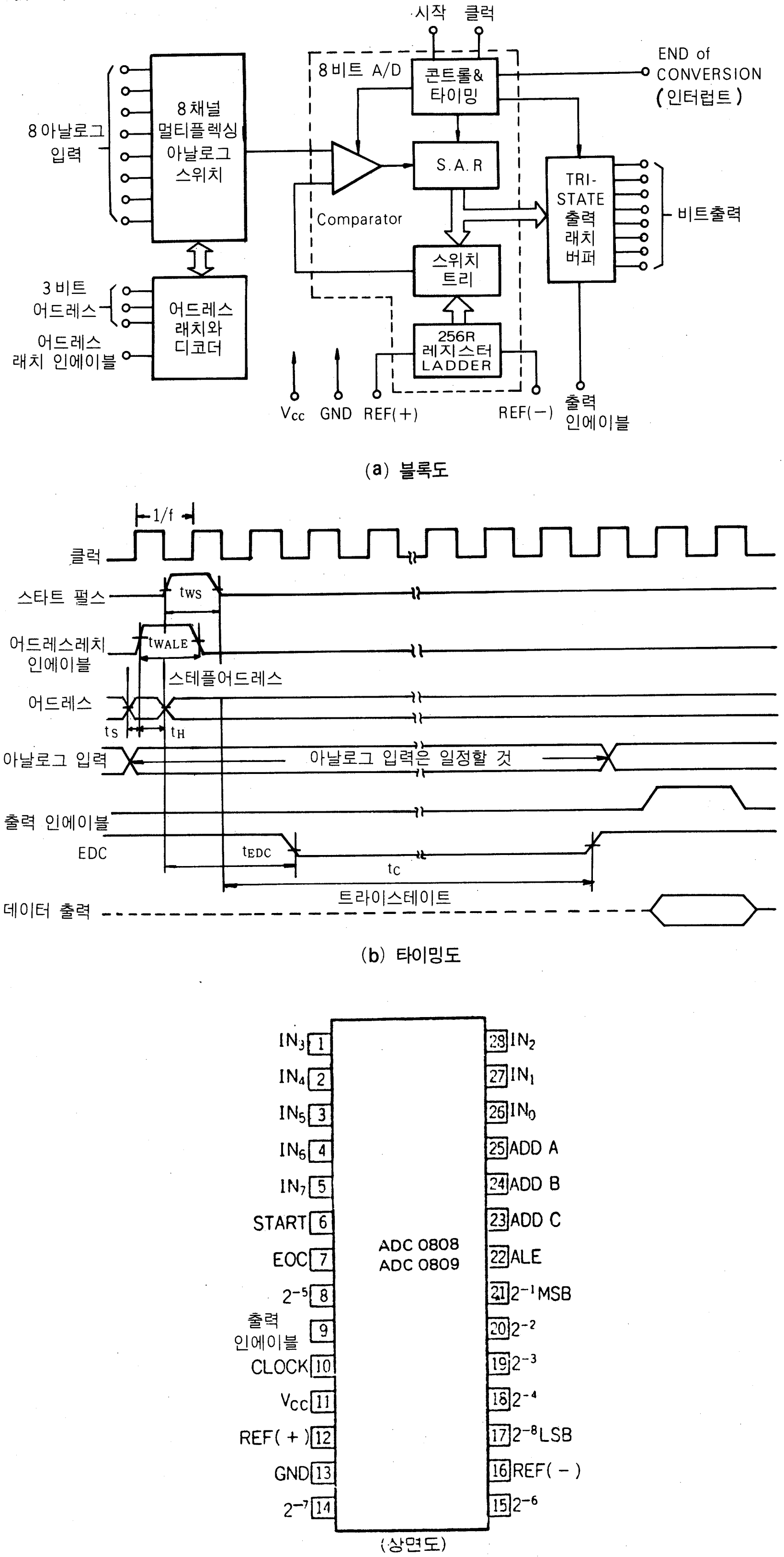


마지막단은 바로 ADC0809이다. 7번 핀의 EOC는 읽지 않았다. 이 신호는 변환 종료신호로서 'H'인 경우 변환이 종료되었다는 것을 나타낸다. 6핀의 START 와 22번핀의 ALE를 붙여 놓았으므로 변환신호를 줌과 동시에 채널이 선택된다. 채널은 23~25핀을 접지시켜 놓았기 때문에 26번핀의 IN_0가 항상 선택된다. 10번핀으로는 약 640kHz의 클럭이 공급되어야 한다. 신호는 이미 만들어 놓은 발진기로 부터 얻을 수 있다. 전체 변환 시간은 100μsec인데 이정도면 이번 작품에 쓰기에는 충분하다. 속도가 빠른 것도 많이 있지만 가격이 갑자기 비싸진다.

D－A 변환부의 제작

D－A 변환기는 디지탈신호를 입력으로 하여 아날로그신호를 출력으로 꺼내는 회로이다. 최근의 컴퓨터는 대부분 디지탈형식의 것이기 때문에 어떤 정보를 처리하는 경우 반드시 디지탈로 변환할 필요가 있다. 그러나 처리된 정보를 출력하는 경우는 아날로그로 하는 것이 보다 효율적인 경우가 많다. 만약 콤팩트 디스크에서 나오는 신호를 소리가 아닌 숫자로 표시한다면 어떨까? 이렇듯 디지탈신호를 아날

〔그림 12〕 ADC 0809



(a) 블록도

(b) 타이밍도

(c) 핀 배치도

로그신호로 변환하여야 할 경우 D−A 변환기가 필요한 것이다.

D−A 변환기는 일반적으로 A−D 변환기에 비하여 매우 간단하다. 가격적인 면에서도 훨씬 싼 것이 보통이다. 또 저항을 적당히 조합하여 쉽게 만들수도 있다.

그림 14에 DAC0808의 블록다이어그램과 핀배치를 그림 15에 회로도를 보였다. 회로가 매우 간단하기 때문에 저역필터와 앰프를 같은 기판에 꾸미는 것도 괜찮다. 배선에 문제가 없다면 틀림 없이 동작할 것이다. 전체회로도는 그림 16과 같다.

소프트웨어

이번 제작에 사용된 프로그램은 리스트 1과 같다. 조금 긴 편이고 같은 문장의 반복이 많기 때문에 틀리지 않도록 주의해야 한다. 최종단계에서는 디버거로 조작하기 때문에 나중에 수정하기는 매우 어려우므로 미리 주의할 필요가 있다. 우선 프로그램을 작성하는데 필요한 화일들은 아래와 같다.

· 에디터:EDLIN. COM

필자의 경우 터보파스칼을 사용하였다.

· MASM. EXE
· DEBUG. EXE
· LINK. EXE
· EXE2BIN. EXE

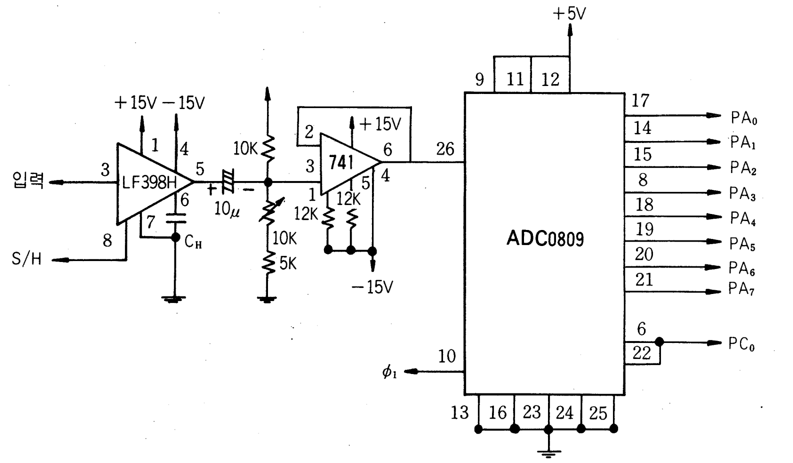
위의 화일이 준비되었으면 에디터를 써서 리스트 1의 프로그램을 작성하도록 하자. 모두 작성하였으면 아래의 과정에 따라 T. COM화일을 만들도록 한다.

```
A>MASM CLOCK, ,CLOCK ; ⏎
A>LINK CLOCK ; ⏎
A>EXE2BIN  CLOCK  T.COM ⏎
```

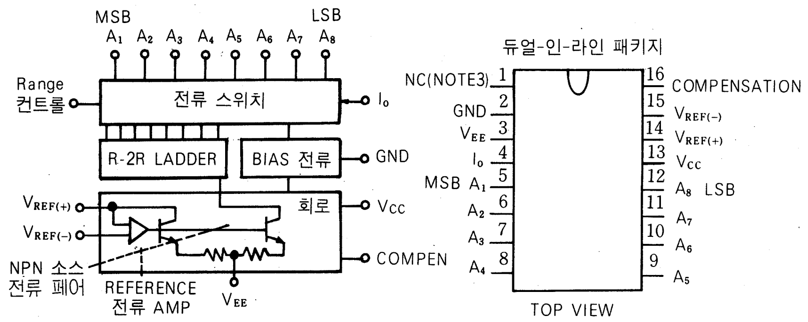
만약 에러가 없다면 다음 순서로 디버거 상으로 들어가야 한다.

```
A>DEBUG ⏎
```

〔그림 13〕 A-D 변환부



〔그림 14〕 DAC 0808




```
- NT.COM ↵
- L ↵
```

이제 음성을 입력할 모든 준비가 되었다. 입력할 음성의 종류는

「지금시각은, 오전, 오후, 열두시, 한시, 두시, 세시, 네시, 다섯시, 여섯시, 일곱시, 여덟시, 아홉시, 열시, 열한시, 십, 이십, 삼십, 사십, 오십, 분입니다, 일, 이, 삼, 사, 오, 육, 칠, 팔, 구」

이다. 이것을 약20초 안에 모두 입력시켜야 한다. 따라서 조금 빨리 입력시키도록 몇번가량 연습을 하기 바란다. 그리고 톤이 일정하도록 하여야 나중에 어색하지 않을 것이다. 조금 낮은 목소리로 또박또박 입력시키는 것이 한번에 성공하는데 유리하다. 하지만 수정할 기회는 얼마든지 있으니까 너무 두려워할 필요는 없다. 자 이제 시작해보자.

```
- G ↵
```

리턴키를 치자마자 입력을 시작하도록 한다. 20초가 지난 후에는 입력한 목소리가 스피커로 들릴 것이다. 만약 모든 음성을 입력시키지 못했거나 입력한 음성이 마음에 들지않으면 다음 과정을 반복하도록 한다.

```
- NT.COM ↵
- L ↵
- G ↵
```

입력이 모두 끝났으면 프로그램의 수정을 시작해야 한다. 우선 프로그램을 T.COM화일에 다시 세이브 시키자.

```
- RBX ↵
BX 0000
: ↵
- RCX ↵
CX 0000
: F100 ↵
- W ↵
```

RBX와 RCX는 세이브된 프로그램의 길이를 정하는 역할을 하고 W는 실제 세이브시키는 역할을 한다.

프로그램을 짜면서

```
MOV  BX , 0F000H
MOV  SI , 500H
```

란 부분이 매우 많음을 느꼈을 것이다.

```
MOV  BX , 0F000H
```

는 데이터의 길이를 세트하는 부분이고

```
MOV  SI , 500H
```

는 데이터의 시작번지를 나타내는 부분이다. 이것을 풀이하면 SI에 해당되는 메모리번지로부터 BX의 값에 해당되는 만큼의 데이터를 선택한다는 의미가 된다. 예를 들어

```
MOV  BX , 1000
MOV  SI , 1300
CALL DACONV
```

는 1300번지부터 1000만큼의 데이터를 D-A 변환한다는 의미가 된다. 실제로 실험해보자. 먼저 V명령을 써서 프로그램의 내용을 보자.

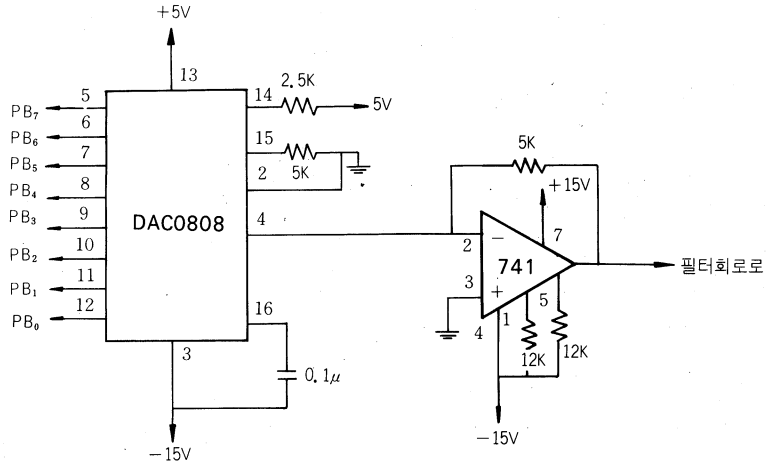
```
- V100 ↵
```

이렇게 하면 첫행이 다음과 같이 출력된다.

```
0EAA : 0100  8CC8  MOV  AX , CS
```

아마 첫 「0EAA」부분은 같지 않을 것이다. 이것은 아무래도 상관없다. 중요한 것은「0100」부분이다. 앞으로는 이것을 번지라고 부르기로 하겠다. 디버거상에서는 어셈블러와 달리 모든 숫자가 16진수로 표시됨을 주의해야

〔그림 15〕 D-A 변환부



한다. 따라서 16진수"A"의 경우 AH 라 쓸 필요가 없고 단지 A라고만 쓰면 된다. 반면에 어셈블러의 경우는 끝에 반드시 "H"를 붙여야 하고 앞에는 변수와 구별하기 위해 숫자 0을 붙여야 한다.

이제 100번지 이하로 계속된 프로그램을 계속보자. 필자의 소프트웨어와 동일하게 작성된 것이라면 010F번지에

```
JMP 0329
```

라고 되어 있을 것이다. 우선 이것을 031C번지로 점프하도록 바꿔야 한다.

```
- A010F ⏎
0EAA :010F  JMP  031C ⏎
0EAA :0112 ⏎
-V100 ⏎
```

아마 010F번지의 내용이 예상대로 바뀌어져 있을 것이다.

```
-G ⏎
```

프로그램에 문제가 없다면 처음에 입력한 내용이 그대로 스피커로 출력되어 나올 것이다.

이상이 없다면 BX의 값과 SI의 값을 변환시켜보기 바란다.

```
- NT.COM ⏎
```

```
- L ⏎
-V31C ⏎
-A31C ⏎
0EAA:031C  MOV  BX , 2000 ⏎
0EAA:031F  MOV  SI , 1500 ⏎
0EAA:0322 ⏎
- G ⏎
```

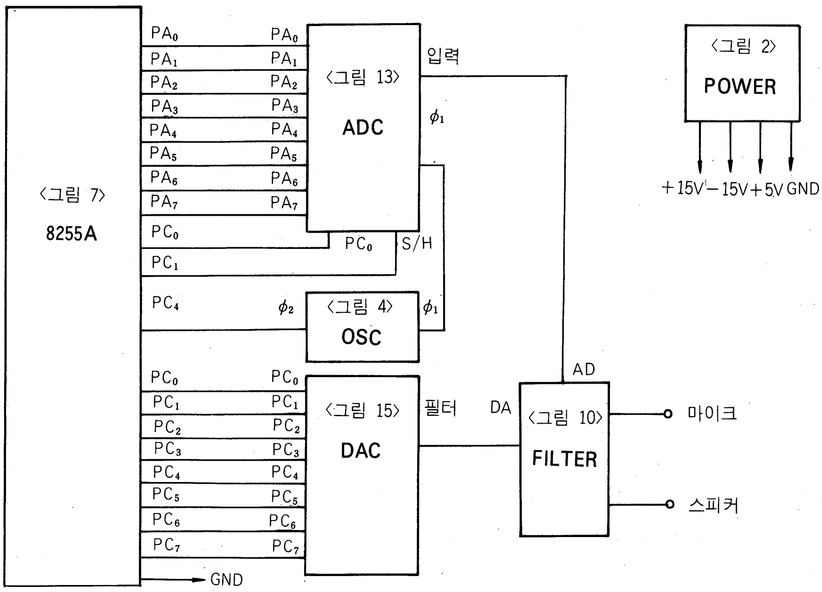
이번엔 입력된 음성 중의 어느 부분인가가 잠시 출력되어 나올 것이다. 이렇게 하여 각 음소들이 저장된 메모리 번지와 그 길이를 하나하나 찾아나가야 한다. 찾아야할 음소들은 처음에 입력한것 즉,"지금시각은","오전"등으로 찾아나가되 "분입니다" 에서 "입니다"를 더 끄집어내기 바란다.

찾아야할 모든 음소(총31개)를 다 찾았으면 이들을 전부 프로그램에 기록해야 한다. 아마 지금까지의 내용을 이해하였다면 프로그램에 기록하는 방법은 다 알았을 것이다. 그러나 하나의 예를 들어 다시한번 설명하기로 하겠다.

우선 CLOCK.LST의 내용을 보자. 가장 왼쪽에는 번지가 나와 있다. 이것을 디버거 때와 비교해보면 꼭 100H 번지 만큼의 차이가 날 것이다. 따라서 리스트 화면에서의 0020번지는 디버거상에서 0120번지를 의미한다. 그러면, "오전"이란 음소가 기록될 위치는「before:」 라고 라벨이 붙은 부분이므로 002E번지, 디버거상에서는 012E번지임을 알 수 있다. 이렇게 각 음소가 기록될 위치를 찾으면 된다. 참고로「pre:」에는"지금시각은"이「first:」에는 "한시"가「j2:」에는"이십"이 기록된다. (표 1 참고)

이제,"오전"이란 음소의 위치를 프

〔그림 16〕 전체 전속도



〔표 1〕 제작에 사용된 음소표

라벨	음 소	라벨	음 소
pre	지금시각은	j2	이십
before	오전	j3	삼십
after	오후	j4	사십
twelveth	열두시	j5	오십
first	한시	zero	입니다
second	두시	one	일
third	세시	two	이
forth	네시	three	삼
fifth	다섯시	four	사
sixth	여섯시	five	오
seventh	일곱시	six	육
eighth	여덟시	seven	칠
ninth	아홉시	eight	팔
tenth	열시	nine	구
eleventh	열한시	last	분입니다
j1	십		

로그램에 실제로 기록하여 보자. 우선 DEBUG부터 기동시켜야 한다.

```
A>DEBUG ↵
- NT.COM ↵
- L ↵
- A012E
0EAA:012E  MOV  BX , 1000 ↵
0EAA:0131  MOV  SI , 1300 ↵
0EAA:0134 ↵
- RBX ↵
BX 0000
: ↵
- RCX ↵
CX 4100
:F100 ↵
- W ↵
```

자, 이것으로 끝났다. 나머지 음소들을 같은 방법으로 모두 기록하기 바란다. 그리고 다음과 같이 010F번지의 JMP 031C를 NOP로 바꾼 후 프로그램을 실행시켜 보자.

```
- A10F ↵
0EAA:010F NOP ↵
0EAA:0110 NOP ↵
0EAA:0111 NOP ↵
0EAA:0112 ↵
```

단, 우선은 세이브를 시키기 바란다. 모두 되었으면 MS-DOS상으로 돌아가 T를 실행시켜 보자.

```
A>T ↵
```

제작을 마치면서

자세히 쓰려고 노력은 하였지만 지식과 경험의 부족으로 결국 실패한 듯 싶다. 한가지 위안을 얻고 싶다면 제작은 실행착오를 하면서 얻은 것이 한번에 성공하는 것보다 훨씬 크고 많다는 사실이다. 아마 대부분의 경우 한번에 성공하기는 어려울 것이다. 이때는 뒤에 소개하는 참고도서를 참고하기 바란다.

많은 것을 얻을 수 있을 것이다. PC

〈참고문헌〉

1.Operational Amplifiers and Integrated Ciruit, Coughlin · Driscoll, Prentice-Hall
2.Electronic Fundamentals and Applications, Ryder, Mcgraw-Hill
3.D-A, A-D 인터페이스 기술, 편집부, 세운
4.Design of OP-Amp Circuit with experiments, Berlin, Howard W.Sams
5.Digital Logic and Computer Design, Mano, Prentice-Hall
6.IBM-PC 하드웨어, IBM, 기전연구사
7.MS-DOSver 3.20, 대우통신
8.MS-DOS 매크로 어셈블러, 황희융, 교학사
9.MS-DOS 매크로 어셈블러 활용1, 황희융, 교학사
10.IBM PC 어셈블리, 신동준, 기전연구사
11.마이컴시스템설계 KNOWHOW, 편집부, 세운
12.OP 앰프 IC활용 KNOWHOW, 편집부, 세운
13.Z-80 마이컴 회로 마스터, 이해동, 청암
14.전자작품해설, 이해동, 청암
15.세계 TTL/IC 규격호환표, 기전연구사
16.최신 산업용 리니어IC규격표, 편집부, 세운
17.최신 AD/DA 콘버터규격표, 편집부, 세화
18.최신 C-MOS IC 규격표, 편집부, 세운

〔리스트 1〕 음성출력시계 소스 리스트

```
Microsoft (R) Macro Assembler  Version 4.00            11/19/88 13:06:20

0000                        main     segment
                                     assume   cs:main,ds:main
= 01F3                      cm       equ      01f3h
= 01F0                      pa       equ      01f0h
= 01F1                      pb       equ      01f1h
= 01F2                      pc       equ      01f2h
0000  8C C8                          mov      ax,cs
0002  8E D8                          mov      ds,ax
0004  8C C8                          mov      ax,cs
0006  8E C0                          mov      es,ax
0008  BA 01F3                        mov      dx,cm
```

```
000B  B0 98                  mov    al,98h
000D  EE                     out    dx,al
000E  FC                     cld
000F  E9 0229 R              jmp    start1
0012  BB F000      pre:      mov    bx,0f000h
0015  BE 0500                mov    si,500h
0018  E8 02BF R              call   daconv
001B  B4 2C        time1:    mov    ah,2ch
001D  CD 21                  int    21h
001F  8A D5                  mov    dl,ch
0021  8A E9                  mov    ch,cl
0023  80 FA 0C     time2:    cmp    dl,12
0026  7C 06                  jl     before
0028  80 EA 0C               sub    dl,12
002B  EB 0D 90               jmp    after
002E  BB F000      before:   mov    bx,0f000h
0031  BE 0500                mov    si,500h
0034  E8 02BF R              call   daconv
0037  EB 0A 90               jmp    time3
003A  BB F000      after:    mov    bx,0f000h
003D  BE 0500                mov    si,500h
0040  E8 02BF R              call   daconv
0043  80 FA 00     time3:    cmp    dl,0
0046  74 35                  je     twelveth
0048  80 FA 01               cmp    dl,01
004B  74 3C                  je     first
004D  80 FA 02               cmp    dl,02
0050  74 43                  je     second
0052  80 FA 03               cmp    dl,03
0055  74 4A                  je     third
0057  80 FA 04               cmp    dl,04
005A  74 51                  je     forth
005C  80 FA 05               cmp    dl,05
005F  74 58                  je     fifth
0061  80 FA 06               cmp    dl,06
0064  74 5F                  je     sixth
0066  80 FA 07               cmp    dl,07
0069  74 66                  je     seventh
006B  80 FA 08               cmp    dl,08
006E  74 6D                  je     eighth
0070  80 FA 09               cmp    dl,09
0073  74 74                  je     ninth
0075  80 FA 0A               cmp    dl,10
0078  74 7B                  je     tenth
007A  E9 0101 R              jmp    eleventh
007D  BB F000      twelveth:mov     bx,0f000h
0080  BE 0500                mov    si,500h
0083  E8 02BF R              call   daconv
0086  E9 010D R              jmp    time4
0089  BB F000      first:    mov    bx,0f000h
008C  BE 0500                mov    si,500h
008F  E8 02BF R              call   daconv
0092  EB 79 90               jmp    time4
0095  BB F000      second:   mov    bx,0f000h
0098  BE 0500                mov    si,500h
009B  E8 02BF R              call   daconv
009E  EB 6D 90               jmp    time4
00A1  BB F000      third:    mov    bx,0f000h
00A4  BE 0500                mov    si,500h
00A7  E8 02BF R              call   daconv
00AA  EB 61 90               jmp    time4
00AD  BB F000      forth:    mov    bx,0f000h
00B0  BE 0500                mov    si,500h
00B3  E8 02BF R              call   daconv
00B6  EB 55 90               jmp    time4
00B9  BB F000      fifth:    mov    bx,0f000h
00BC  BE 0500                mov    si,500h
00BF  E8 02BF R              call   daconv
00C2  EB 49 90               jmp    time4
00C5  BB F000      sixth:    mov    bx,0f000h
00C8  BE 0500                mov    si,500h
00CB  E8 02BF R              call   daconv
00CE  EB 3D 90               jmp    time4
00D1  BB F000      seventh:  mov    bx,0f000h
00D4  BE 0500                mov    si,500h
00D7  E8 02BF R              call   daconv
00DA  EB 31 90               jmp    time4
00DD  BB F000      eighth:   mov    bx,0f000h
00E0  BE 0500                mov    si,500h
00E3  E8 02BF R              call   daconv
00E6  EB 25 90               jmp    time4
00E9  BB F000      ninth:    mov    bx,0f000h
00EC  BE 0500                mov    si,500h
00EF  E8 02BF R              call   daconv
00F2  EB 19 90               jmp    time4
00F5  BB F000      tenth:    mov    bx,0f000h
00F8  BE 0500                mov    si,500h
00FB  E8 02BF R              call   daconv
00FE  EB 0D 90               jmp    time4
0101  BB F000      eleventh:mov     bx,0f000h
0104  BE 0500                mov    si,500h
0107  E8 02BF R              call   daconv
010A  EB 01 90               jmp    time4
010D  80 FD 0A     time4:    cmp    ch,10
0110  7C 62                  jl     time5
0112  80 ED 0A               sub    ch,10
0115  80 FD 0A               cmp    ch,10
0118  7C 1E                  jl     j1
011A  80 ED 0A               sub    ch,10
011D  80 FD 0A               cmp    ch,10
0120  7C 22                  jl     j2
0122  80 ED 0A               sub    ch,10
0125  80 FD 0A               cmp    ch,10
0128  7C 26                  jl     j3
```

```
012A  80 ED 0A                 sub    ch,10
012D  80 FD 0A                 cmp    ch,10
0130  7C 2A                    jl     j4
0132  80 ED 0A                 sub    ch,10
0135  EB 31 90                 jmp    j5
0138  BB F000         j1:      mov    bx,0f000h
013B  BE 0500                  mov    si,500h
013E  E8 02BF R                call   daconv
0141  EB 31 90                 jmp    time5
0144  BB F000         j2:      mov    bx,0f000h
0147  BE 0500                  mov    si,500h
014A  E8 02BF R                call   daconv
014D  EB 25 90                 jmp    time5
0150  BB F000         j3:      mov    bx,0f000h
0153  BE 0500                  mov    si,500h
0156  E8 02BF R                call   daconv
0159  EB 19 90                 jmp    time5
015C  BB F000         j4:      mov    bx,0f000h
015F  BE 0500                  mov    si,500h
0162  E8 02BF R                call   daconv
0165  EB 0D 90                 jmp    time5
0168  BB F000         j5:      mov    bx,0f000h
016B  BE 0500                  mov    si,500h
016E  E8 02BF R                call   daconv
0171  EB 01 90                 jmp    time5
0174  80 FD 00        time5:   cmp    ch,0
0177  74 2B                    je     zero
0179  80 FD 01                 cmp    ch,1
017C  74 32                    je     one
017E  80 FD 02                 cmp    ch,2
0181  74 39                    je     two
0183  80 FD 03                 cmp    ch,3
0186  74 40                    je     three
0188  80 FD 04                 cmp    ch,4
018B  74 47                    je     four
018D  80 FD 05                 cmp    ch,5
0190  74 4E                    je     five
0192  80 FD 06                 cmp    ch,6
0195  74 55                    je     six
0197  80 FD 07                 cmp    ch,7
019A  74 5C                    je     seven
019C  80 FD 08                 cmp    ch,8
019F  74 63                    je     eight
01A1  EB 6D 90                 jmp    nine
01A4  BB F000         zero:    mov    bx,0f000h
01A7  BE 0500                  mov    si,500h
01AA  E8 02BF R                call   daconv
01AD  EB 76 90                 jmp    last1
01B0  BB F000         one:     mov    bx,0f000h
01B3  BE 0500                  mov    si,500h
01B6  E8 02BF R                call   daconv
01B9  EB 61 90                 jmp    last
01BC  BB F000         two:     mov    bx,0f000h
01BF  BE 0500                  mov    si,500h
01C2  E8 02BF R                call   daconv
01C5  EB 55 90                 jmp    last
01C8  BB F000         three:   mov    bx,0f000h
01CB  BE 0500                  mov    si,500h
01CE  E8 02BF R                call   daconv
01D1  EB 49 90                 jmp    last
01D4  BB F000         four:    mov    bx,0f000h
01D7  BE 0500                  mov    si,500h
01DA  E8 02BF R                call   daconv
01DD  EB 3D 90                 jmp    last
01E0  BB F000         five:    mov    bx,0f000h
01E3  BE 0500                  mov    si,500h
01E6  E8 02BF R                call   daconv
01E9  EB 31 90                 jmp    last
01EC  BB F000         six:     mov    bx,0f000h
01EF  BE 0500                  mov    si,500h
01F2  E8 02BF R                call   daconv
01F5  EB 25 90                 jmp    last
01F8  BB F000         seven:   mov    bx,0f000h
01FB  BE 0500                  mov    si,500h
01FE  E8 02BF R                call   daconv
0201  EB 19 90                 jmp    last
0204  BB F000         eight:   mov    bx,0f000h
0207  BE 0500                  mov    si,500h
020A  E8 02BF R                call   daconv
020D  EB 0D 90                 jmp    last
0210  BB F000         nine:    mov    bx,0f000h
0213  BE 0500                  mov    si,500h
0216  E8 02BF R                call   daconv
0219  EB 01 90                 jmp    last
021C  BB F000         last:    mov    bx,0f000h
021F  BE 0500                  mov    si,500h
0222  E8 02BF R                call   daconv
0225  B4 4C           last1:   mov    ah,4ch
0227  CD 21                    int    21h
0229  BB F000         start1:  mov    bx,0f000h
022C  BF 0500                  mov    di,500h
022F  BA 01F2                  mov    dx,pc
0232  EC              search1: in     al,dx
0233  24 10                    and    al,10h
0235  75 FB                    jnz    search1
0237  EC              search2: in     al,dx
0238  24 10                    and    al,10h
023A  74 FB                    je     search2
023C  B0 02           repeat1: mov    al,02h
023E  EE                       out    dx,al
023F  B0 03                    mov    al,03h
0241  EE                       out    dx,al
0242  B0 00                    mov    al,00h
0244  EE                       out    dx,al
```

```
0245 EC          search3: in    al,dx
0246 24 10                and   al,10h
0248 75 FB                jnz   search3
024A EC          search4: in    al,dx
024B 24 10                and   al,10h
024D 74 FB                je    search4
024F BA 01F0              mov   dx,pa
0252 EC                   in    al,dx
0253 24 F0                and   al,11110000b
0255 8A E0                mov   ah,al
0257 BA 01F2     repeat2: mov   dx,pc
025A B0 02                mov   al,02h
025C EE                   out   dx,al
025D B0 03                mov   al,03h
025F EE                   out   dx,al
0260 B0 00                mov   al,00h
0262 EE                   out   dx,al
0263 EC          search5: in    al,dx
0264 24 10                and   al,10h
0266 75 FB                jnz   search5
0268 EC          search6: in    al,dx
0269 24 10                and   al,10h
026B 74 FB                je    search6
026D BA 01F0              mov   dx,pa
0270 EC                   in    al,dx
0271 24 F0                and   al,11110000b
0273 B1 04                mov   cl,4h
0275 D2 C8                ror   al,cl
0277 0A C4                or    al,ah
0279 AA                   stosb
027A 4B                   dec   bx
027B BA 01F2              mov   dx,pc
027E 75 BC                jnz   repeat1
0280 EB 01 90             jmp   start2
0283 BB F000     start2:  mov   bx,0f000h
0286 BE 0500              mov   si,500h
0289 EC          search7: in    al,dx
028A 24 10                and   al,10h
028C 75 FB                jnz   search7
028E EC          search8: in    al,dx
028F 24 10                and   al,10h
0291 74 FB                je    search8
0293 BA 01F1     repeat3: mov   dx,pb
0296 AC                   lodsb
0297 8A E0                mov   ah,al
0299 24 F0                and   al,11110000b
029B EE                   out   dx,al
029C BA 01F2              mov   dx,pc
029F EC          search9: in    al,dx
02A0 24 10                and   al,10h
02A2 75 FB                jnz   search9
02A4 EC          search10:in    al,dx
02A5 24 10                and   al,10h
02A7 74 FB                je    search10
02A9 BA 01F1     repeat4: mov   dx,pb
02AC 8A C4                mov   al,ah
02AE 24 0F                and   al,00001111b
02B0 B1 04                mov   cl,4h
02B2 D2 C0                rol   al,cl
02B4 EE                   out   dx,al
02B5 4B                   dec   bx
02B6 BA 01F2              mov   dx,pc
02B9 75 CE                jne   search7
02BB B4 4C                mov   ah,4ch
02BD CD 21                int   21h
02BF 52          daconv:  push  dx
02C0 BA 01F2              mov   dx,pc
02C3 EC                   in    al,dx
02C4 24 10                and   al,10h
02C6 75 F7                jnz   daconv
02C8 EC          daconv1: in    al,dx
02C9 24 10                and   al,10h
02CB 74 FB                je    daconv1
02CD BA 01F1     daconv2: mov   dx,pb
02D0 AC                   lodsb
02D1 8A E0                mov   ah,al
02D3 24 F0                and   al,11110000b
02D5 EE                   out   dx,al
02D6 BA 01F2              mov   dx,pc
02D9 EC          daconv3: in    al,dx
02DA 24 10                and   al,10h
02DC 75 FB                jnz   daconv3
02DE EC          daconv4: in    al,dx
02DF 24 10                and   al,10h
02E1 74 FB                je    daconv4
02E3 BA 01F1     daconv5: mov   dx,pb
02E6 8A C4                mov   al,ah
02E8 24 0F                and   al,00001111b
02EA B1 04                mov   cl,4h
02EC D2 C0                rol   al,cl
02EE EE                   out   dx,al
02EF 4B                   dec   bx
02F0 BA 01F2              mov   dx,pc
02F3 75 CA                jne   daconv
02F5 5A                   pop   dx
02F6 C3                   ret
02F7             main     ends
                          end
```

```
Microsoft (R) Macro Assembler  Version 4.00   11/19/88 13:06:20
                                               Symbols


Segments and Groups:

                N a m e                    Size    Align   Combine
                                                           Class

MAIN . . . . . . . . . . . . . .           02F7    PARA    NONE

Symbols:

                N a m e                    Type    Value   Attr

AFTER  . . . . . . . . . . . . .           L NEAR  003A    MAIN

BEFORE . . . . . . . . . . . . .           L NEAR  002E    MAIN

CM . . . . . . . . . . . . . . .           Number  01F3

DACONV . . . . . . . . . . . . .           L NEAR  02BF    MAIN
DACONV1  . . . . . . . . . . . .           L NEAR  02C8    MAIN
DACONV2  . . . . . . . . . . . .           L NEAR  02CD    MAIN
DACONV3  . . . . . . . . . . . .           L NEAR  02D9    MAIN
DACONV4  . . . . . . . . . . . .           L NEAR  02DE    MAIN
DACONV5  . . . . . . . . . . . .           L NEAR  02E3    MAIN

EIGHT  . . . . . . . . . . . . .           L NEAR  0204    MAIN
EIGHTH . . . . . . . . . . . . .           L NEAR  00DD    MAIN
ELEVENTH . . . . . . . . . . . .           L NEAR  0101    MAIN

FIFTH  . . . . . . . . . . . . .           L NEAR  00B9    MAIN
FIRST  . . . . . . . . . . . . .           L NEAR  0089    MAIN
FIVE . . . . . . . . . . . . . .           L NEAR  01E0    MAIN
FORTH  . . . . . . . . . . . . .           L NEAR  00AD    MAIN
FOUR . . . . . . . . . . . . . .           L NEAR  01D4    MAIN

J1 . . . . . . . . . . . . . . .           L NEAR  0138    MAIN
J2 . . . . . . . . . . . . . . .           L NEAR  0144    MAIN
J3 . . . . . . . . . . . . . . .           L NEAR  0150    MAIN
J4 . . . . . . . . . . . . . . .           L NEAR  015C    MAIN
J5 . . . . . . . . . . . . . . .           L NEAR  0168    MAIN

LAST . . . . . . . . . . . . . .           L NEAR  021C    MAIN
LAST1  . . . . . . . . . . . . .           L NEAR  0225    MAIN

NINE . . . . . . . . . . . . . .           L NEAR  0210    MAIN
NINTH  . . . . . . . . . . . . .           L NEAR  00E9    MAIN

ONE  . . . . . . . . . . . . . .           L NEAR  01B0    MAIN

PA . . . . . . . . . . . . . . .           Number  01F0
PB . . . . . . . . . . . . . . .           Number  01F1
PC . . . . . . . . . . . . . . .           Number  01F2
PRE  . . . . . . . . . . . . . .           L NEAR  0012    MAIN

REPEAT1  . . . . . . . . . . . .           L NEAR  023C    MAIN
REPEAT2  . . . . . . . . . . . .           L NEAR  0257    MAIN
REPEAT3  . . . . . . . . . . . .           L NEAR  0293    MAIN
REPEAT4  . . . . . . . . . . . .           L NEAR  02A9    MAIN

SEARCH1  . . . . . . . . . . . .           L NEAR  0232    MAIN
SEARCH10 . . . . . . . . . . . .           L NEAR  02A4    MAIN
SEARCH2  . . . . . . . . . . . .           L NEAR  0237    MAIN
SEARCH3  . . . . . . . . . . . .           L NEAR  0245    MAIN
SEARCH4  . . . . . . . . . . . .           L NEAR  024A    MAIN
SEARCH5  . . . . . . . . . . . .           L NEAR  0263    MAIN
SEARCH6  . . . . . . . . . . . .           L NEAR  0268    MAIN
SEARCH7  . . . . . . . . . . . .           L NEAR  0289    MAIN
SEARCH8  . . . . . . . . . . . .           L NEAR  028E    MAIN
SEARCH9  . . . . . . . . . . . .           L NEAR  029F    MAIN
SECOND . . . . . . . . . . . . .           L NEAR  0095    MAIN
SEVEN  . . . . . . . . . . . . .           L NEAR  01F8    MAIN
SEVENTH  . . . . . . . . . . . .           L NEAR  00D1    MAIN
SIX  . . . . . . . . . . . . . .           L NEAR  01EC    MAIN
SIXTH  . . . . . . . . . . . . .           L NEAR  00C5    MAIN
START1 . . . . . . . . . . . . .           L NEAR  0229    MAIN
START2 . . . . . . . . . . . . .           L NEAR  0283    MAIN

TENTH  . . . . . . . . . . . . .           L NEAR  00F5    MAIN
THIRD  . . . . . . . . . . . . .           L NEAR  00A1    MAIN
THREE  . . . . . . . . . . . . .           L NEAR  01C8    MAIN
TIME1  . . . . . . . . . . . . .           L NEAR  001B    MAIN
TIME2  . . . . . . . . . . . . .           L NEAR  0023    MAIN
TIME3  . . . . . . . . . . . . .           L NEAR  0043    MAIN
TIME4  . . . . . . . . . . . . .           L NEAR  010D    MAIN
TIME5  . . . . . . . . . . . . .           L NEAR  0174    MAIN
TWELVETH . . . . . . . . . . . .           L NEAR  007D    MAIN
TWO  . . . . . . . . . . . . . .           L NEAR  01BC    MAIN

ZERO . . . . . . . . . . . . . .           L NEAR  01A4    MAIN


    317 Source  Lines
    317 Total   Lines
     85 Symbols

  48524 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors
```

전2중 1200bps용 카드형 모뎀

최근들어 PC통신에 대한 일반인들의 관심이 높아지면서 인텔리전트기능을 내장한 내장카드형 모뎀들이 등장하였다.

일반인들의 관심을 불러 일으키는 이 카드들은 다음과 같은 구조와 기능을 가지고 있다.

인텔리전트 모뎀의 기능

인텔리전트 모뎀은 간단한 코맨드를 입력하는 것만으로 데이터통신을 할 때에 필요한 각종의 접속동작을 자동적으로 하여준다. 따라서 최근에는 PC의 대량보급과 함께 많은 관심을 불러 일으키고 있다.

이와 같은 인텔리전트 모뎀의 기본적인 구성은 **그림 1**에서 보는 바와 같이 4부분으로 되어 있다.

●코맨드 인터프리터부

먼저 인텔리전트 모뎀의 가장 큰 특징은 코맨드 인터프리터라고 하는 기능을 내장하고 있는 것에 있다. 이 코맨드 인터프리터는 PC 등의 데이터단말(DTE)에서 입력된 코맨드의 의미를 해석하고 모뎀내부의 각 기능에 구체적인 동작을 시키는 것이며, 그 내부는 **그림 2**에서와 같은 구성으로 되어 있다.

여기서 그림 2에 있어서의 각부의 동작을 간단하게 설명하면 다음과 같다.

먼저 코맨드 인터프리터부에서는 코맨드인식부에서 코맨드의 시작코드를 검출하면 이후 종료코드가 검출되기까지 입력된 데이터를 코맨드 버퍼라고 하는 메모리기능에 격납한다. 그리고 코맨드입력의 종료와 함께 코맨드해설부가 이 버퍼의 내용에 대해서 그 정당성을 체크하면서 순차 구체적인 내부명령으로 변환하여 간다.

따라서 만일 이 디코드시에 있어서 버퍼의 내용에 잘못된 기술이 있으면 내부명령의 생성은 되지 않고 에러메시지를 입력코맨드에 대한 실행결과로서 응답코드생성부에서 출력한다.

또 모뎀내부의 각 기능(모뎀부나 NCU부)은 미리 메모리부에 격납된 각종의 파라미터를 참조하면서 이 내부명령에 따른 일련의 동작을 실행하므로 그 동작완료시의 상황에서 입력된 코맨드에 대한 응답코드가 출력된다.

그림 3은 이상의 동작을 나타내는 순서도인데 이와 같은 코맨드 인터프리터의 동작에 따라 DTE측에서는 모뎀내부의 상세한 동작을 의식하는 일 없이 손쉽게 데이터통신을 할 수 있다.

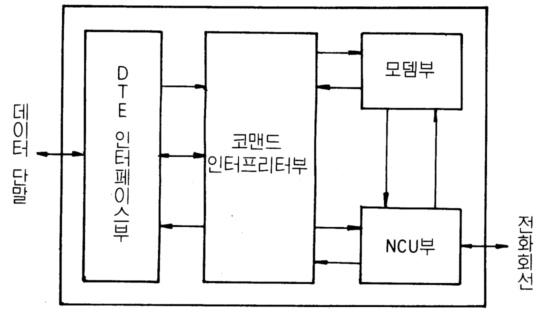
●DTE 인터페이스부

DTE인터페이스부는 모뎀과 DTE를 접속하기 위한 기능을 가지고 있으며 이하에 나타내는 항목에 대해서 그 인터페이스조건이 규정되어 있다.

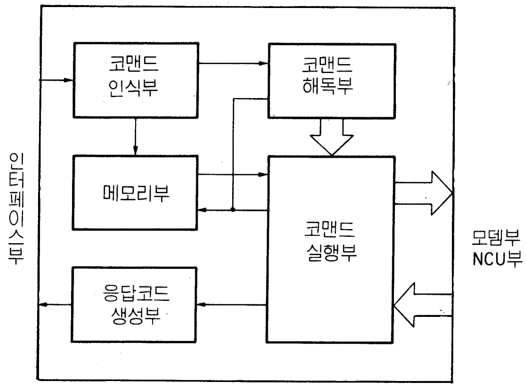
- **기계적특성** : 코넥터형상, 치수, 핀배열 등
- **전기적특성** : 신호전압레벨, 입출력임피던스, 신호검출레벨 등
- **논리적특성** : 상호접속회로의 기능정의 동작조건, 제어순서 등

이러한 인터페이스 조건은 모뎀의 형

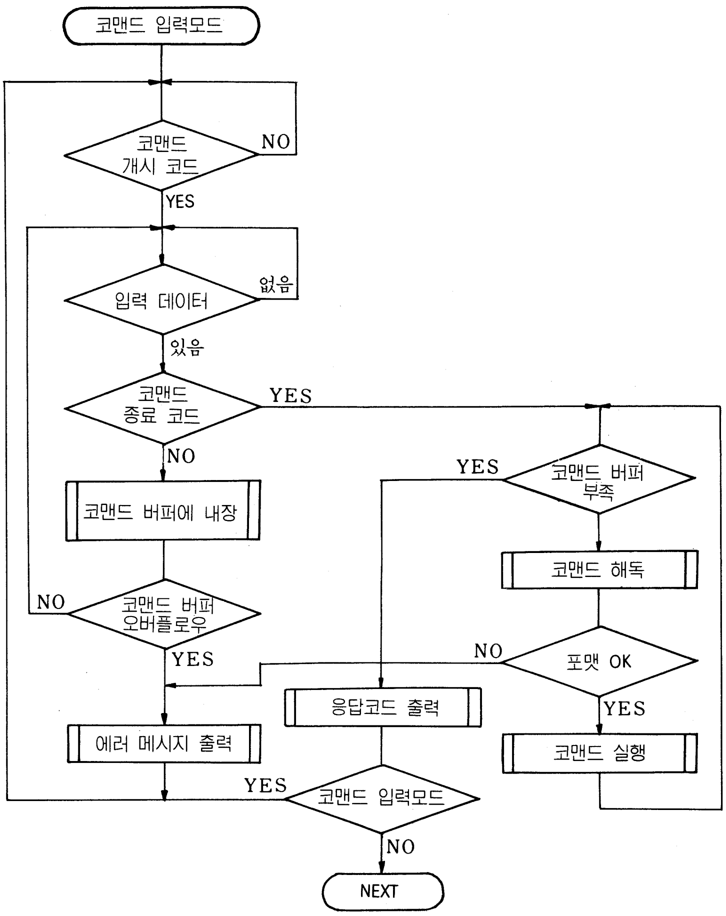
〔그림 1〕 인텔리전트 모뎀의 구성



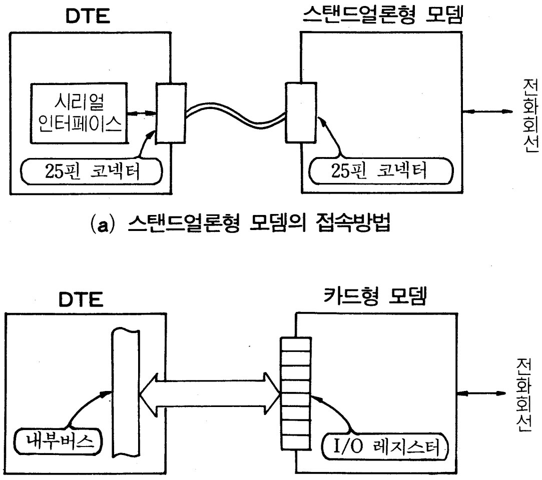
〔그림 2〕 코맨드 인터프리터부의 구성



〔그림 3〕 코맨드 인터프리터의 동작 순서도



〔그림 4〕 모뎀의 접속방법



(a) 스탠드얼론형 모뎀의 접속방법

(b) 카드형 모뎀의 접속방법

태에 따라 다르며 모뎀의 접속방법은 크게 나누어 2가지로 구분할 수 있다.

그림 4(a)는 스탠드 얼론형의 모뎀을 DTE에 접속하는 방법이며, 통상 25핀의 코넥터를 거쳐서 신호의 주고받음이 이루어진다. **표 1**은 이와 같은 접속방법에 적용되는 일반적인 인터페이스규격을 그 규정내용에 대해서 정리한 것이다.

표 1에 있어서의 CCITT권고는 EIA(미국전자공업회)의 RS-232C규격을 국제판으로 한 것이다. 예를 들면 25핀 코넥터에 대한 기능정의에 대해서 RS-232C규격을 표로 나타내면 **표 2**와 같다.

그림 4(b)는 모뎀을 DTE의 내부버스 등에 접속하는 방법이며, 일반적으로 이런 종류의 모뎀은 그 형상에서 카드형 모뎀이라 하고 있다. 카드형 모뎀의 경우 DTE와의 신호의 주고받음은 어드레스제어된 I/O레지스터에의 라이트와 리드에 의해서 이루어진다.

따라서 그 접속부에 있어서의 인터페이스조건은 DTE의 버스구조나 시스템이 서포트하는 시리얼 인터페이스의 구성에 따라서 다르며, 현재까지의 시점에서 표 1에 나타내는 바와 같은 일반규격은 없다.

그러나 PC통신이 발달된 미국에서는 대부분이 카드형 모뎀이 주류로 되어 있으며, 그 이유는 IBM-PC의 설계사상이 강하게 반영된 것과 모뎀이 PC의 내부기능이라고 여기는 사상때문이라고 한다.

IBM-PC는 주지하는 바와 같이 세계에서 가장 보급율이 높은 16비트 퍼스널컴퓨터이지만 동일한 환경하에서 여러개의 시리얼 포트를 유저가 정의할 수 있다. 현재 국내에서 입수가능한 IBM-PC 호환기종의 대부분은 표준장비되어 있는 하드웨어적인 시리얼 포트가 없으며 모뎀 등을 접속할 경우에는 별도의 시리얼 포트를 카드식으로 내장하여야 한다.

●모뎀부와 NCU부

모뎀부는 전화회선을 거쳐서 실제로 2진데이터를 전송하기 위한 기능이며, 데이터통신에 있어서의 데이터의 투과성을 보증하기 위해서는 그 기능 및 특성이 일반의 모뎀규격에 일치할 필요가 있다.

NCU(회선제어장치)부는 모뎀을 전화회선에 접속하기 위한 회선인터페이스 기능이다. 일반적으로 전화회선에 단말설비(모뎀 등)을 접속하는 경우에는 각국의 국내법에 근거하는 기술적인 기준이 있다.

기술기준의 내용에 대해서는 나라에 따라서도 다소 다르다. 그 목적으로 하는 것은 인체에의 안전과 공공통신설비의 보호이기 때문에 단말설비는 반드시 이 내용을 준수하는 것이 의무로 되어 있다.

〔표 1〕 모뎀 인터페이스의 일반 규격

규 정 내 용		ISO규격 (CCITT 권고)	공업규격 EIA
물리적 특성		ISO IS2110	RS-232C
전기적 특성		CCITT V.10 CCITT V.11 CCITT V.28	
논리적 특성	기능정의	CCITT V.24	
	동작조건	CCITT V.25, V·26bis (NCU기능) CCITT V.54(시험기능)	
	제어순서	일반 모뎀규격(CCITT규격, BELL규격)	

NCU부에 속하는 중요한 회로인 하이브리드회로와 링신호검출회로에 대해서 설명을 한다.

●하이브리드회로

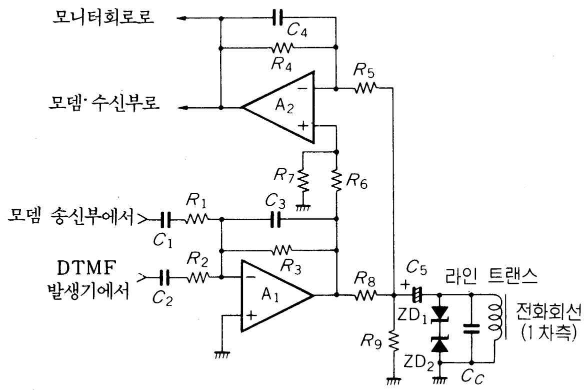
2선식의 전화회선에 모뎀을 접속하는 경우에는 모뎀송신부에서의 신호에너지는 전화회선에, 또 통신상대로 부터의 신호에너지는 모뎀수신부에 효율좋게 전달하는 것이 필요하다. 그래서 **그림 5**에 이와 같은 2선-4선변환에 사용되는 하이브리드회로의 구성 예를 나타낸다.

그림 5의 하이브리드회로에 있어서의 각 소자는 사용주파수대에서 앞서 설명한 신호에너지가 가장 효율좋게 전달되도록 설정된다. 또 라인 트랜스포머의 삽입손실을 보상하는 경우에는 그림속의 R_3와 R_4를 조정하면 된다.

●링신호검출회로

링검출신호는 통상 **그림 6**에 나타내는 바와 같이 전화회선상의 교류신호를 제너 다이오드나 포토 아이솔레이터 등에서 포락검파함으로써 얻을 수 있다.

〔그림 5〕 하이브리드회로



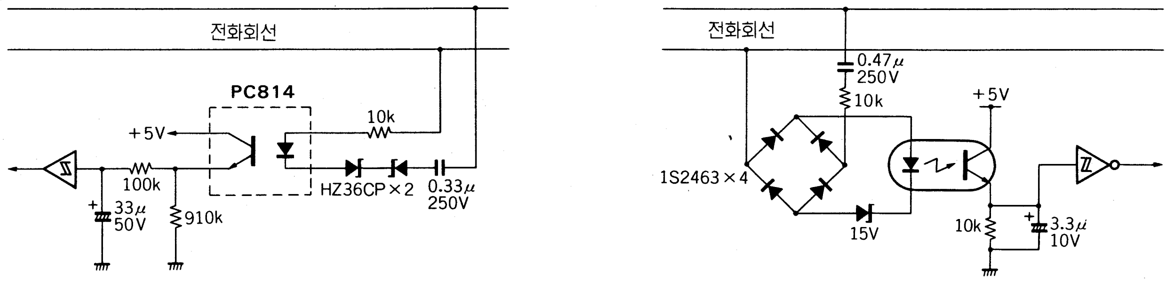
어느 나라의 기술기준에서도 단말설비가 전화회선에 직류를 공급하는 것은 엄하게 금지되어 있다. 따라서 단말설비의 회선입력부에 있어서 라인 트랜스포머나 릴레이 또는 포토 커플러 등에 의한 완전한 전원분리가 의무로 되어 있으며 어스회로에 대해서도 마찬가지로 별도 배선으로 한다.

또 이 분리의 경계선에서 회선측(1차측)의 회로부품에 대해서는 고전압발생시에 있어서도 특성열화가 없는 것을 사용하는 것이 필요하다. 또한 통상 전화회선에는 낙뢰발생시의 서어지전압이나 전파 등의 사용대역외 잡음이 혼입할 가능성이 있으며, NCU부에는 이와 같은 회선잡음에 대한 방지기능도 필요하게 된다.

〔표 2〕 25핀 코넥터에 있어서의 기능정의표

핀번호	신호명칭 EIA RS-232C	회 로 명 칭	핀번호	신호명칭 EIA RS-232C	회 로 명 칭
1	AA	보안용 접지	14	SBA	백워드 채널 송신 데이터
2	BA	송신 데이터	15	DB	송신신호 엘레먼트 타이밍(DCE)
3	BB	수신 데이터	16	SBB	백워드 채널 수신 데이터
4	CA	송신 요구	17	DD	수신신호 엘레먼트 타이밍(DCE)
5	CB	송신 가능	18		
6	CC	데이터 세트 레디	19	SCA	백워드 채널 송신 요구
7	AB	신호용 접지	20	CD	데이터 단말 레디
8	CF	데이터 채널 수신 캐리어 검출	21	CG	데이터 신호 품질
9					
10			22	CE	피호출 표시
11			23	CH/CI	데이터신호 속도 선택
12	SCF	백워드 채널 수신 캐리어 검출	24	DA	송신신호 엘레먼트 타이밍(DTE)
13	SCB	백워드 채널 송신 가능	25		

〔그림 6〕 링신호 검출회로의 예



(a) 제너 다이오드에 의한 링신호 검출회로 (b) 다이오드 브리지에 의한 링신호 검출회로

IBM-PC용 모뎀보드의 하드웨어

여기서는 세이코엡슨제의 인텔리전트 모뎀용 LSI인 STC9610F를 사용하여 IBM-PC용 카드형 모뎀보드를 설계하기로 한다.

●모뎀보드의 개요

이 모뎀보드의 규격은 **표 3**에 나타내는 바와 같이 지금까지 설명하여 온 기능을 조합한 것이며 IBM-PC용이라는 것에서 DTE인터페이스는 IBM-PC의 버스 사양으로 되어 있다. 또 코맨드에 대해서는 현재 이런 종류의 모뎀에서는 표준적인 「AT코맨드」를 채용하고 있다. **그림 7**은 이 모뎀보드의 블록구성을 나타낸 것이지만 인텔리전트 모뎀에 필요한 기능은 NCU부를 제외하고 모두 IC화 되어 있다.

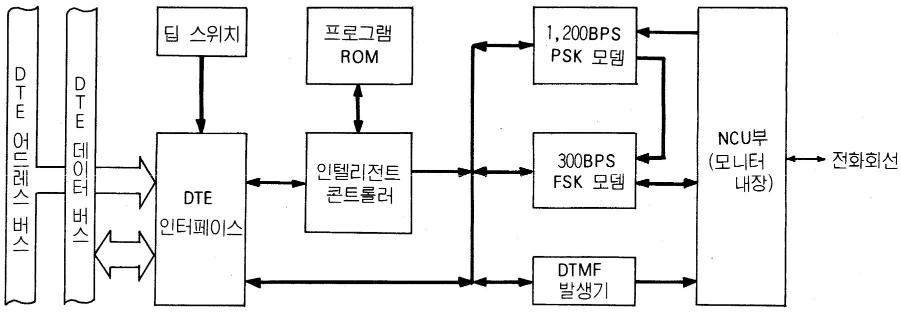
●각부의 구성

그림 8은 IBM-PC의 확장슬롯에 직

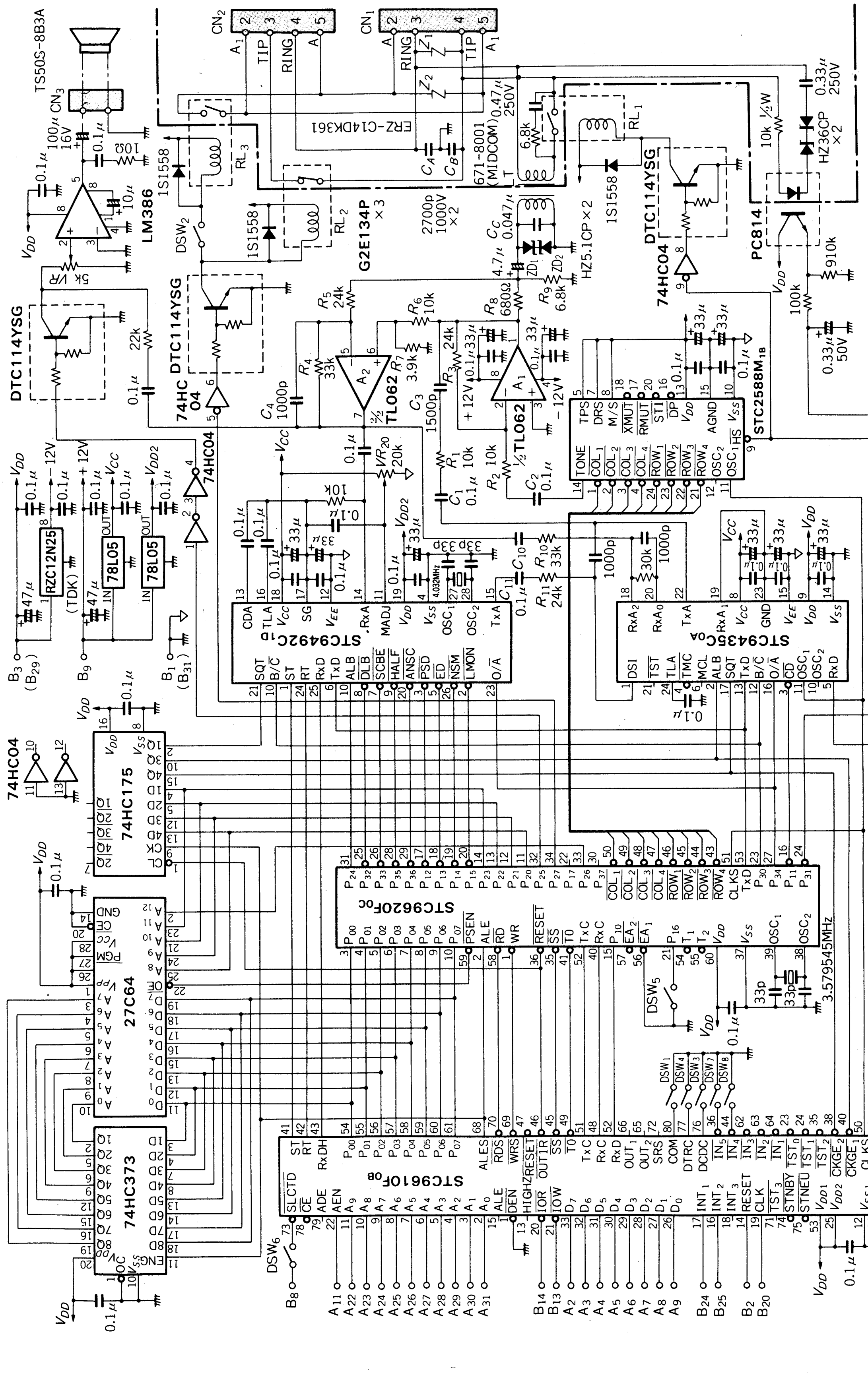
〔표 3〕 모뎀보드의 규격

기 능	항 목	규 격
DTE 인터페이스	물리적특성	62핀 카드 에지 코넥터
	전기적특성	IBM-PC XT/AT 버스규격
	논리적특성	
	데이터 포맷	스톱 비트…… 1비트 데이터 비트… 9, 10비트 패리티 비트… 우수, 기수, 고정, 없음 스톱 비트…… 1비트이상
	DTE간 데이터 속도	110bps, 300bps(조보식) 1,200bps+1/−2.5%(조보식)
코맨드 인터프리터	코맨드체계	Hayes사 AT 코맨드 호환
	코맨드 버퍼용량	최대 40문자
모 뎀	통신방식	전이중/반이중
	회선 데이터 속도	110bps, 300bps(조보식) 1,200bps±0.01(동기식)
	적합 모뎀 규격	CCITT V.21, BELL 103J(300bps) CCITT V.22, BELL 212A(1,200bps)
NCU	회선접속	(발신) 자동/수동 선택 가능 (착신) 자동/수동 선택 가능
	다이얼신호	펄스 다이얼방식 10pps, 20pps 톤 다이얼방식 DTMF신호
	회선 단말 인터페이스	600Ω±30%
	회선 모니터기능	스피커 내장(음량 조절 가능)
기 타	테스트 기능	아날로그, 루프백 테스트
	사용 전원전압	+5V, +12V

〔그림 7〕 IBM-PC용 모뎀보드의 블록도



〔그림 8〕 IBM-PC용 모뎀보드의 전회로도



접 접속이 가능한 카드형 모뎀의 전체 회로도이며, 여기서 사용한 LSI와 각 부의 구성에 대해서 설명한다.

IBM-PC의 비동기통신용 시리얼 포트에 대해서는 여러개의 독립한 채널을 설정하는 것이 가능하며 그 각 채널에 대해서 표 4에 나타내는 I/O레지스터가 정의되어 있다.

이 보드에서 사용한 STC9610F$_{OB}$는 이와 같은 IBM-PC와의 버스 인터페이스에 필요한 기능을 내장한 LSI이며 그림 9에 그 내부블록구성을 나타낸다.

또 이 보드에 있어서의 코맨드 인터프리터기능은 8비트의 전용프로세서인 STC9620F의 펌웨어에 의해서 구성하고 있다. 또 STC9620F의 주변기능은 그림 10에 나타내는 바와 같이 타이밍 신호를 제외하고 모두 내부의 레지스터를 거쳐서 접속하고 있으며, 앞서 설명한 STC6910F$_{OB}$도 포트0(P_{00}~P_{07})의 외부데이터 메모리로 되어 있다.

모뎀부는 변복조용의 신호처리LSI인 STC9492C$_{ID}$와 STC9435C$_{OA}$를 사용하여 1,200bps와 300bps의 전2중모뎀을 구성하고 있다. 이 LSI는 어느 것이나 CCITT규격과 BELL규격의 선택이 가능하기 때문에 이 모뎀보드는 CCITT V.21/V.22 및 BELL 103J/212A의 모뎀과 데이터통신을 할 수 있다.

〔표 4〕 I/O 레지스터의 어드레스 대응표

I/O 어드레스	DLAB 비트	I/O 레지스터	Read/Write
XF8	0	송신보유 레지스터	W
XF8	0	수신 버퍼 레지스터	R
XF8	1	분주래치 레지스터(LS)	R/W
XF9	0	인터럽트 인에이블 레지스터	R/W
XF9	1	분주래치 레지스터(MS)	R/W
XFA	*	인터럽트 식별 레지스터	R
XFB	*	회선제어 레지스터	R/W
XFC	*	모뎀제어 레지스터	R/W
XFD	*	회선상황 레지스터	R
XFE	*	모뎀상황 레지스터	R
XFF	*	스크래치 레지스터	R/W

주) IBM-PC XT의 경우 COM1은 X=3, COM2는 X=2이다.

그림 8의 오른쪽에 있는 RL$_1$은 직류루프를 개폐하는 오프후크 릴레이이며, RL$_2$는 RL$_1$과 함께 전화회선의 전환을 하는 릴레이이다. 또 RL$_3$은 특수한 용도에 사용되는 릴레이이며 일반의 2선식 전화회선에서는 없어도 상관없다.

회선입력부에 있는 Z$_1$과 C$_A$, C$_B$ 및 라인트랜스포머 LT의 2차측에 있는 ZD$_1$, ZD$_2$와 C$_C$는 회선상에서 발생한 서어지전압이나 전파의 혼입에 의한 대역외잡음의 제거를 목적으로 하고 있다.

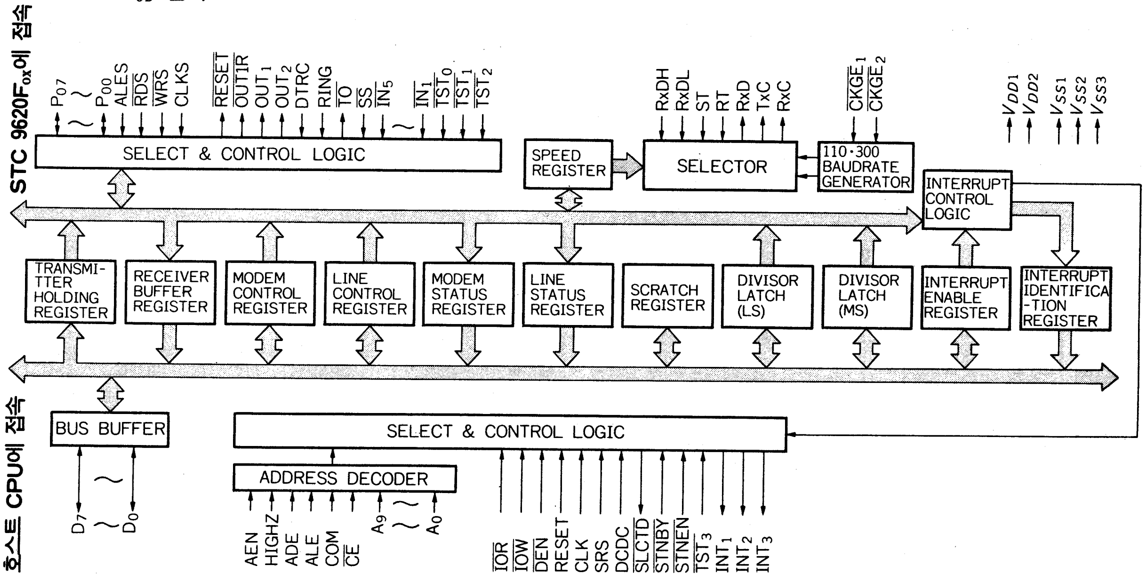
또 톤다이얼에 필요한 DTMF신호는 STC2588M$_{IB}$라고 하는 DTMF발생용 IC의 출력신호를 하이브리드회로의 송신앰프 A$_1$을 거쳐서 전화회선에 송출하도록 하고 있다.

이 보드상에는 이 외에 부가기능으로서 그림 11에 나타내는 바와 같은 스피커에 의한 회선모니터회로가 있다. 회선모니터회로는 하이브리드회로의 수신앰프출력을 단순히 전력증폭할 뿐이며, 볼륨에 의한 스피커의 음량조정과 코맨드에 의한 스피커의 ON/OFF제어가 가능하다.

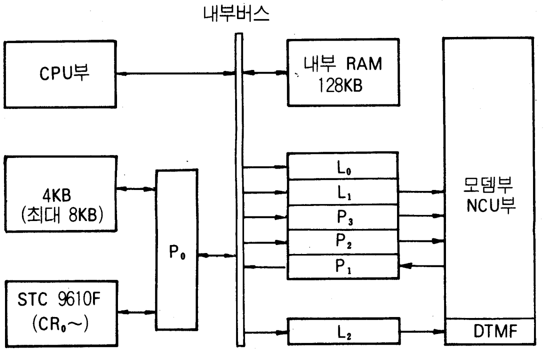
●제작상의 주의점

앞에서 설명한 바와 같이 단말설비에 대해서는 각국의 국내법에 의해 기술기준을 준수하는 것이 의무로 되어 있다.

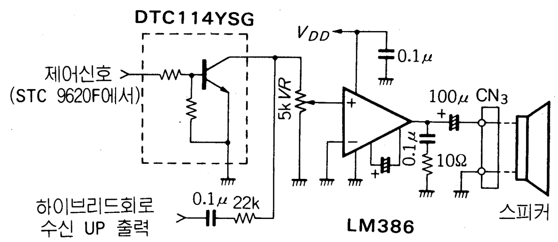
〔그림 9〕 STC 9610F$_{OB}$ 블록도



〔그림 10〕 STC 9620F의 내부 레지스터 구성도



〔그림 11〕 회선 모니터회로



그래서 특히 NCU의 회선입력부(그림 8의 쇄선테두리부분)에 대해서는 다른 회로와 다소 떨어져 배치한다.

또 서어지전압방지용 콘덴서 C_A, C_B는 다른 어스회로와는 별도로 배선하여 접지저항이 100Ω 이하의 보안용 어스(케이스어스)에 접속하여야 한다. 그림 8에 나타내는 회로도에서는 +5V의 전원계통을 3개로 나누어 별도 배선으로 하고 있다.

이것은 스위치용의 캐패시터회로를 내장하고 있는 LSI에 대해서 전원잡음의 혼입을 방지하는 것이 목적이다. 그 외에 일반적인 사항으로서

(ㄱ) 전원라인의 배선저항은 될 수 있는 한 낮게 억제한다.
(ㄴ) 아날로그신호선과 디지탈신호선을 서로 섞이게 하지 않는다.
(ㄷ) IC의 단자에 접속하는 콘덴서는 단자의 가까이에 배치한다.
(ㄹ) 큰 과도전류가 발생하는 회로(예를 들면 릴레이 등)의 전원은 별도 배선으로 한다.
(ㅁ) 그라운드 루프를 만들지 않는다.

등의 점을 보드제작시에는 주의하여야 한다.

● 레벨 조정의 방법

전체회로도에서는 콜프로그레스톤(발신음, 통화중음 등)의 검출레벨을 조정하기 위해 20kΩ의 가변저항 VR_{20}이 있다. 또 모뎀의 송신캐리어나 DTMF톤의 송출레벨 및 상대모뎀에서의 캐리어검출레벨 등을 조정하는 경우가 상정된다. 그래서 이들의 조정순서와 조정방법에 대해서 이하 간단히 설명한다.

① 송신신호의 레벨조정

(ㄱ) 회선접속코넥터(CN_1)에 종단임피던스 600Ω의 레벨계를 접속한다(그림 12(a) 참조).
(ㄴ) 뒤에 있는 샘플 프로그램에서 통신속도를 설정한 후 터미널모드에서 오프후크후에(ATH1 ↲ 을 입력한다) 다음의 코맨드를 입력하여 강제적으로 송신캐리어를 발생시킨다.

ATC1D ↲ (송신모드)
ATC1A ↲ (수신모드)

(ㄷ) 레벨계의 표시값이 설정레벨로 되도록 각 저항값을 조정한다. 설정레벨에 대해서는 우리나라의 국내 기준에 따라야 한다. 저항값의 조정순서는 우선 FSK캐리어(300bps모드)에 대해서 R_1을 조정후 다음에 PSK캐리어(1,200bps모드)를 R_{11}에서 조정한다.

② DTMF신호의 조정

(ㄱ) ①의 경우와 같은 측정회로를 설정한다.
(ㄴ) 샘플 프로그램의 터미널모드에서 온후크후 ATS16=2 ↲ 를 입력한다.
(ㄷ) "OK" 또는 "0"이 응답코드로서 확인된 후 임의의 키입력을 하면 DTMF신호가 발생한다.
(ㄹ) 레벨계의 표시값이 우리나라의 국내 기준에 정해진 범위로 되도록 R_2를 조정한다.

그후 키입력을 할 때마다 순차 DTMF주파수가 변화하므로 모든 경우에 대해서 레벨 체크를 하여야 한다.

③ 수신캐리어검출레벨의 조정

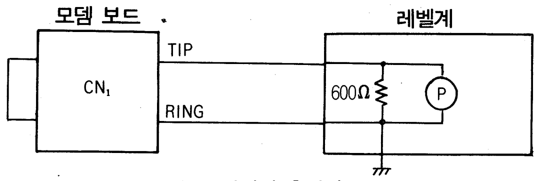
(ㄱ) 회선접속용 코넥터(CN_1)에 신호원 임피던스가 600Ω인 발진기를 접속한다(그림 12 참조).
(ㄴ) 샘플 프로그램에서 소정의 통신모드를 설정한 후 발진기의 주파수를 대응하는 모뎀규격의 중심주파수로 설정한다.
(ㄷ) AT코맨드로 오프후크한 후 레벨계(입력임피던스 10kΩ 이상)를 보면서 600Ω평형형 아테네이터로 −50dBm 정도로 수신신호레벨을 조정한다.
(ㄹ) 이 상태에서 다음의 코맨드를 입력한다.

ATD ↲ (송신모드)
ATA ↲ (수신모드)

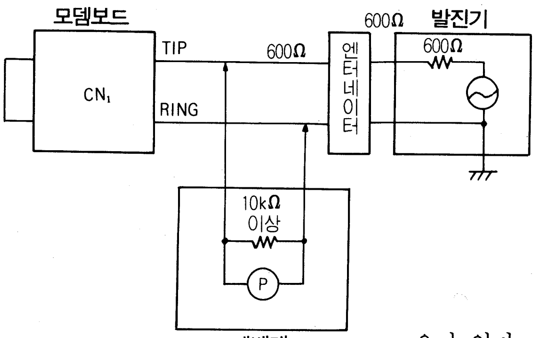
이대로는 모뎀보드는 아무 것도 응답하지 않는다.

(ㅁ) 아테네이터로 수신신호레벨을 모뎀이 톤 또는 송신캐리어를 발생하기까지 서서히 크게 하여 간다.
(ㅂ) 이상의 조작을 반복해서 하여 모뎀이 응답하는 수신신호레벨이 각 모뎀규격에서 정해진 값을 만족하도록 각 저항값을 조정한다(일반적으로 모뎀규격에서는 수신캐리어신호에 대해서 −43dBm~−45dBm 정도가 권고되고 있다). 저항값의 조정순서는 우

〔그림 12〕 송수신 레벨의 측정

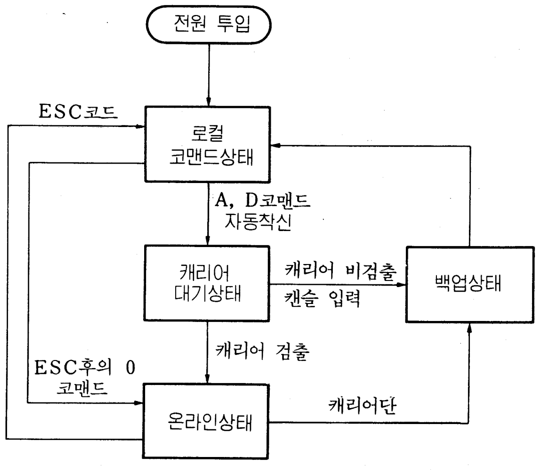


(a) 송신레벨 측정회로



(b) 수신레벨 측정회로

〔그림 13〕 모뎀의 상태도



선 1,200bps모드에 대해서 R_4 또는 R_5를 조정한 후 300bps모드에서 R_{10}을 조정한다.

④ 콜프로그레스톤 검출레벨의 조정

(ㄱ) 수신캐리어 검출레벨의 조정후 발진기의 주파수를 400Hz 정도로 설정한다.

(ㄴ) AT코맨드로 오프후크한 후 아테네이터에서 모뎀보드로의 입력신호레벨이 −50dBm로 되도록 조정한다.

(ㄷ) 이 상태에서 다음의 AT코맨드를 입력하여 다이얼신호가 발생하지 않는 것을 확인한다.

ATX2 ↲
ATDT123 ↲

(ㄹ) 다음에 입력신호레벨을 −35dBm~−40dBm로 설정하여 (ㄷ)와 마찬가지의 코맨드를 입력하였을 경우 다이얼신호가 발생하도록 VR_{20}을 조정한다.

AT코맨드

모뎀을 동작시키기 위해서는 모뎀을 IBM-PC에 삽입하여 코맨드를 줄 필요가 있다. 코맨드에는 크게 나누어 2가지의 체계가 있으며, 하나는 CCITT 권고의 V.25bis, 또 하나는 미국의 모뎀메이커 Hayes사가 고안한 AT코맨드이다.

AT코맨드는 V.25bis에 비해서 이해하기 쉽고, 체계적으로도 확고하게 되어 있기 때문에 미국에서는 압도적인 지지를 얻고 있다. 현재 시판되고 있는 IBM-PC용의 통신소프트웨어의 거의가 AT코맨드대응으로 되어 있는 것으로 부터도 그 우위성을 알 수 있다.

●모뎀의 상태변화

AT코맨드의 구체적인 설명을 하기 전에 모뎀의 변화에 대해서 간단히 설명한다. **그림 13**에 나타낸 이들 상태도 중에서 AT코맨드는 로컬코맨드 상태일 때만 접수된다.

●로컬코맨드 상태

모뎀은 전원투입후 자동적으로 로컬코맨드 상태로 된다. 이 상태에서 AT코맨드의 입력이 가능하다. 모뎀과 모뎀을 접속하기 위해서는 A, D코맨드를 입력하든가, 상대로부터 호출에 의한 자동착신에 의해 캐리어대기 상태로 바뀐다.

●캐리어대기 상태

캐리어대기 상태에서는 상대모뎀과 접속시퀀스를 하여 BELL/CCITT, 1,200/1,300bps의 어느 모드에 접속할 수 있는가를 서로 확인한다.

서로 접속가능한 것이 확인되어 접속이 완료하면 온라인 상태로 바뀐다.

캐리어가 검출되지 않거나 PC에서 취소입력이 있으면 행업(Hang up) 상태로 바뀐다.

●온라인 상태

온라인 상태에서는 상대모뎀과 데이터의 송수신을 할 수 있다. 상대모뎀에서의 캐리어가 일정시간(S10 레지스터에 의한 설정) 없어졌을 경우는 행업상태로 바뀐다.

또 ESC코드가 입력되면 모뎀은 접속된채로 로컬코맨드 상태로 바뀐다. ESC코드는 데이터의 송신중에 잘못 인식되지 않도록 **그림 14**에 나타내는 ESC시퀀스라고 하는 특별한 입력방법이 취해진다.

●행업 상태

행업 상태에서는 모뎀을 회선에서 분리하여 로컬코맨드 상태로 바꾼다.

●AT코맨드의 사용방법

AT코맨드라는 명칭은 코맨드의 선두에 "AT"를 붙이는 것에서 오고 있다.

AT코맨드의 서식은

AT〔코맨드〈파라미터〉〕…{〔코맨드〈파라미터〉〕} 〈CR〉

로 표시되며, "AT"와 〈CR〉(캐리지 리턴)로 둘러싼 문자열이 실제로 실행되는 코맨드로 된다.

코맨드를 기술할 때에는 다음의 사항에 주의할 필요가 있다.

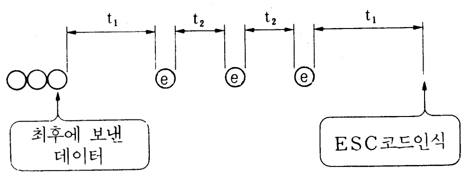
① "AT"는 연속하는 문자열이며 "A"를 입력 후 "T"이외의 문자를 입력하면 "AT"로서 인식되지 않는다. 이와 같은 경우에는 최초의 "A"부터 다시 입력한다.

② 백스페이스에 의해서 입력문자의 소거가 가능하다. 단, "AT"는 ①의 이유로 소거할 수 있다.

③ 파라미터를 생략하였을 경우는 D코맨드를 제외하고 파라미터에 0이 주어진 것으로서 인식된다.

④ 코맨드의 길이는 코맨드 버퍼의 용량(통상 40문자)의 범위내이면 문제없이 접수되지만 코맨드 버퍼의 용량을 넘는 코맨드가 입력되면 오버플로우로 되어 그 코맨드는 실행되지 않고 에러메시지가 되돌아 온다.

〔그림 14〕 ESC 시퀀스



ⓔ : ESC코드 캐릭터(S2 레지스터에서 설정)
$t_1 \geq$ 카드 타임(S12 레지스터에서 설정) $> t_2$

⑤ "AT", 〈CR〉, 스페이스는 코맨드 버퍼에는 격납되지 않는다. 필요에 따라서 스페이스를 삽입함으로써 코맨드의 기술을 보기 쉽게 할 수 있다.

⑥ 코맨드 버퍼에 격납된 코맨드는 〈CR〉의 입력에 의해서 실행된다.

⑦ 코맨드 버퍼에 격납된 코맨드는 다음의 AT코맨드가 입력되기까지 유지된다. 유지되어 있는 코맨드를 재실행하고 싶은 경우는

A/(리피트 코맨드)

를 사용한다. 리피드 코맨드에는 "AT"도 〈CR〉도 필요없다.

●AT코맨드의 기능 설명

코맨드의 종류를 표 5에 나타낸다. 표에서 보는 바와 같이 기본적인 코맨드는 쉽게 이해할 수 있으리라 생각되지만 중요한 A, D코맨드에 대해서는 보충설명을 한다.

●A코맨드

모뎀을 사용하여 통신을 할 경우 우선 송신측과 수신측을 할당할 필요가 있다. 일반적으로 전화를 거는 쪽이 송신, 받는 쪽이 수신측이 된다. 대부분의 시판모뎀에서는 전화의 호출신호에 따라서 자동적으로 응답측을 설정하는 자동착신기능을 가지고 있어서 이용자는 그 할당을 특히 의식할 필요는 없지만 전화로 용건 등을 이야기하고 나서 회선을 끊고 모뎀의 접속을 할 경

〔표 5〕 AT 코맨드 종류

코맨드	파라미터	기 능
A	—	모뎀을 수신모드로 설정
Bn	n=0, 1	0 : CCITT 모드 1 : BELL 모드
Cn	n=0, 1	0 : 캐리어 OFF 1 : 캐리어 ON
Ds	s=0~9 *#( )— , ; TPR	모뎀을 송신모드로 설정 자동 다이얼
En	n=0, 1	0 : 로컬 코맨드 상태에서 ECHO 없음 1 : 로컬 코맨드 상태에서 ECHO 있음
Fn	n=0, 1	0 : 반이중 모드 1 : 전이중 모드
Hn	n=0~2	0 : ON 1 : OFF 2 : 특수 OFF
In	n=0, 1	0 : 버전 No : 참조 1 : 프로그램의 체크섬 참조
Mn	n=0, 1	0 : 모니터 스피커 OFF 1 : 캐리어 검출까지 모니터 스피커 ON 2 : 항상 모니터 스피커 ON

코맨드	파라미터	기 능
O	—	온라인상태로 바꿈
P		자동 다이얼시 펄스 다이얼 선택
Qn	n=0, 1	0 : 응답코드를 되돌린다. 1 : 응답코드를 되돌리지 않는다.
R	—	자동 다이얼시 모뎀을 머신모드로 설정
Sr ? Sr=n	r=0, 17 n : s 레지스터 정의표 참조	Sr ? : S 레지스터의 내용 참조 Sr=n : S 레지스터에 값 n을 설정
T	—	자동 다이얼시 톤 다이얼 선택
Vn	n=0, 1	0 : 응답코드를 숫자로 되돌린다. 1 : 응답코드를 문자로 되돌린다.
Xn	n=0, 1, 2, 3, 4	0 : 기본 응답코드의 선택 1 : 확장 응답코드의 선택 2 : 다이얼 톤만의 검지, 확장 응답코드의 선택 3 : 다이얼 톤만의 검지, 확장 응답코드의 선택 4 : 다이얼 톤, 비지 톤의 검지, 확장 응답코드의 선택
Z	—	모뎀의 리세트

우는 송신측과 수신측을 미리 정하여 둘 필요가 있다. 이런 때 수신측은

ATA〈CR〉

을 입력한다. 한편 송신측은

ATD 〈CR〉

을 입력한다.

●D코맨드

D코맨드에는 A코맨드의 곳에서 설명한 바와 같이 모뎀을 송신측으로 할당하는 동작이 있음과 동시에 자동다이얼을 하는 동작이 있다.

D코맨드의 서식은 "Ds"이며 "s"에는 다이얼 파라미터가 기술된다. 사용가능한 다이얼 파라미터를 이하에 나타낸다.

0~9*#：다이얼번호
T：톤 다이얼선택
P：펄스 다이얼선택
R：모뎀을 수신측으로 전환한다. 상대모뎀이 송신전용모뎀의 경우 등에 사용한다.
( )-：다이얼 파라미터로서는 무시된다. 이 때문에 다이얼번호 기술로서 다음과 같은 입력도 가능하다.

03-111-2222
03(111)2222

,：일정시간(S8 레지스터에 의한 설정) 다이얼동작을 휴지.
;：이후의 다이얼 파라미터는 실행시키지 않고 다음의 AT코맨드 입력을 기다린다.

이상에서 설명한 다이얼 파라미터에서 다음의 자동다이얼도 가능하다.
ATDT0, 03-111-2222 (내선에서 0 발신으로 외선에 접속한다)

●응답코드

응답코드는 RING응답을 제외하고 AT코맨드 실행후에 출력된다(RING응답은 전화의 호출신호에 따라서 출력된다). 응답코드는 숫자로 출력되는 경우와 단어로 출력되는 경우가 있으며, V코맨드에 의해 선택할 수 있다. 응답코드일람을 표 6에 나타낸다.

●S레지스터

모뎀내부에는 모뎀의 동작을 콘트롤하는 파라미터 등을 보관하는 S레지스터가 17개 마련되어 있다. 이들의 레지스터는 S코맨드에 의해서 참조하기도

〔표 6〕 응답코드 정의

숫자	단 어	의 미
0	OK	코맨드행내의 모든 코맨드의 실행이 정상 종료한다. 또는 온라인상태에서 ESC코드를 받아 들인다. →그 다음 코맨드상태로 돌아온다.
1	CONNECT	상대 모뎀과 통신할 수 없는 상태로 된다. X코맨드에 의해 「CONNECT」의 의미가 약간 다르다. X0일 때：300/1,200bps에서 코넥터한다. X1~4일 때：300bps에서 코넥터한다. →그 다음에 온라인상태로 이동한다. 통신 데이터나 ESC코드는 코넥터한 통신속도로 받아들인다.
2	RING	호출 벨이 울린다. 링이 1회 종료하는 동시에 송출된다. →그 다음에 코맨드상태로 돌아온다. 코맨드의 입력중에는 송출되지 않는다.
3	NO CARRIER	지정시간을 대기하여도 캐리어가 검출되지 않는다. 캐리어가 지정시간이상 끊긴다. 통신을 중단한다. →그 다음 코맨드상태로 돌아온다.
4	ERROR	코맨드행에 해석할 수 없는 명령이 있다. 또 코맨드 버퍼(40문자)를 초과한다. →그 다음에 코맨드로 되돌아온다.
5	CONNECT 1200	상대 모뎀과 1,200bps에서 연결한다(단, X코맨드가 X1~4일 때 송출된다.). →그 다음에 온라인상태로 돌아온다. 송신 데이터나 ESC코드는 1,200bps에서 받아들인다.
6	NO TONE	오프후 다이얼 톤을 검출할 수 있다. →온 한 후 코맨드상태로 되돌아온다.
7	BUSY	다이얼 후 비지 톤을 검출한다. →온 한 후 코맨드로 되돌아온다.

〔표 7〕 S레지스터 정의

레지스터 No.	범 위	단 위	초기치	내 용
0	0~255	회	DIPSW 5	자동착신까지의 호출신호(벨)가 반복되는 회수
1	0~255	회	0	호출신호의 반복되는 회수
2	0~127	ASCII	2BH(+)	ESC코드 캐릭터
3	0~127	ASCII	0DH(CR)	캐리지 리턴 코드(복귀)
4	0~127	ASCII	0AH(LF)	라인 피드 코드(개행)
5	0~32, 127	ASCII	08H(BS)	백 스페이스 코드(후퇴)
6	2~255	초	2	오프시킨 후 다이얼 개시까지의 시간
7	0~255	초	30	다이얼 후의 캐리어 대기시간
8	0~255	초	2	휴식시간
9	1~255	1/10초	6	캐리어 검출 확정시간
10	1~255	1/10초	7	캐리어 상실 확정시간 255=캐리어 상실시, 회선 절단 안됨
11	30~255	ms	70	톤 다이얼의 유음시간과 무음시간
12	20~255	1/50초	50	ESC코드의 카드타임
13	비트 맵 레지스터		30H	UART상태 레지스터
14	비트 맵 레지스터		4AH	옵션 레지스터
15	비트 맵 레지스터		3BH	플래그 레지스터
16	0~2, 4	—	0	테스트 모드
17	비트 맵 레지스터		E4H	플래그 레지스터

하고, 설정할 수 있다. 예를 들면 S0 레지스터를 참조할 경우는

ATS0? 〈CR〉

또 S0 레지스터에 "2"를 설정할 경우는

ATS0=2〈CR〉

로 입력한다. S레지스터 종류를 **표 7**에 나타낸다.

모뎀의 사용과 샘플 프로그램

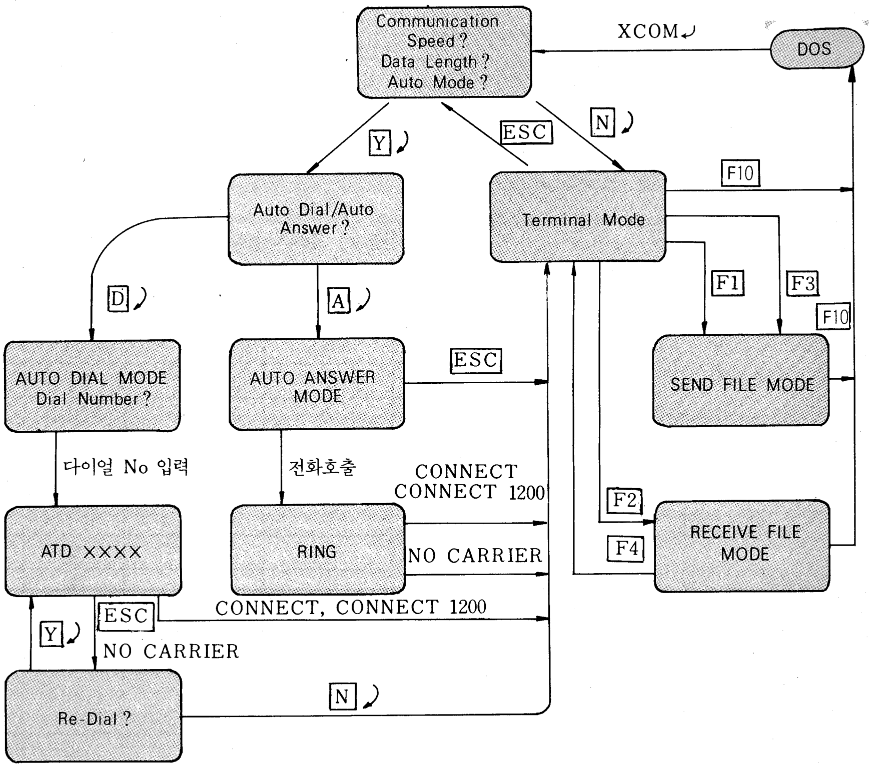
이번에 제작한 모뎀보드에는 8개의 딥스위치가 있다. **표 8**에 그 기능과 샘플 프로그램을 사용할 경우의 설정을 나타낸다.

모뎀을 동작시키는 간단한 통신프로그램을 BASIC으로 작성하였다. 처리속도가 늦을 경우는 컴파일러를 이용하여 컴파일을 할 필요가 있다. 프로그램의 구성은 자동다이얼, 자동착신, 터미널, 화일전송, 화일수신으로 된다. 프로그램 리스트는 **리스트 1**에 나타낸다. 또 조작은 **그림 15**를 참조하기 바란다.

〔표 8〕 딥 스위치의 정의

딥 스위치 번 호	샘플 프로그램 사용시 설정	기 능
1	ON	비동기통신 포트 COM 1/COM 2의 선택 ON : COM 1 OFF : COM 2
2	ON	모듈 잭의 선택 ON : 4선식 OFF : 2선식
3	OFF	Data Carrier Detect(DCD) 콘트롤 ON : 캐리어의 유무에 대응한다. OFF : DCD="1"
4	ON	Data Terminal Ready(DTR)의 모니터 ON : 모니터한다. OFF : 모니터하지 않는다.
5	ON	자동착신의 허가/금지 ON : 금지(S0="0"를 설정) OFF : 허가(S0="2"를 설정)
6	확장 슬롯의 사용 상황에 따라 설정	IBM PC의 확장 슬롯 선택 ON : 슬롯 No. 8(하프 슬롯) OFF : 슬롯 No. 1~7(풀슬롯)
7	가입선에 따라 설정	펄스 다이얼의 펄스 송출속도 ON : 슬롯 No. 8(하프) OFF : 슬롯 No. 1~7(풀)
8	OFF	AI 코맨드의 받아들임의 허가/금지 ON : 금지 OFF : 허가

〔그림 15〕 조작 순서도



〔리스트 1〕 모뎀 제어 프로그램 예

```
10 '****************************************************
20 '*  Auto dial & Auto answer program for IBM PC    *
30 '****************************************************
40 SCREEN 0:CLS:CLOSE
50 KEY OFF
60 KEY 1,"SEND":KEY 2,"RECV":KEY 3, "SNDEND":KEY 4, "RECEND"
   :KEY 5, ""
70 KEY 6, "": KEY 7, "": KEY 8, "": KEY 9, "": KEY 10,"END"
```

```
80 DIM DIALNO$(100): DIM D$(99)
90 XOFF$=CHR$(19): XON$ = CHR$(17)
100 FALSE = 0: TRUE = NOT FALSE
110 ON COM(1) GOSUB 1720
120 ON KEY(1) GOSUB 1870
130 ON KEY(2) GOSUB 1980
140 ON KEY(3) GOSUB 2180
150 ON KEY(4) GOSUB 2240
160 ON KEY(10) GOSUB 2300
170 '*********************************************
180 '*         COMMUNICATION PARAMETER           *
190 '*********************************************
200 INPUT "Communication speed ? (1200,300) ",SP$
210 IF SP$ = "" THEN SP$ = "1200"
220 INPUT "Data length ? (8,7) ",DL$
230 PA$ = "N"
240 IF DL$ = "" THEN DL$ = "8"
250 IF DL$ = "7" THEN PA$ = "E"
260 PO$ = "COM1:"+SP$+","+PA$+","+DL$+",1,RS,CD"
270 OPEN PO$ AS #1
280 INPUT "Auto mode ? (Y,N) ",TR$
290 IF TR$ = "N" OR TR$ = "n" THEN 1340
300 '*********************************************
310 '*              AUTO  MODE                   *
320 '*********************************************
330 PRINT #1,"ATZ"
340 LINE INPUT #1,K$:PRINT K$;
350 IF K$ <> CHR$(&HA)+"OK" THEN 340
360 FOR N = 1 TO 1000:NEXT N
370 PRINT #1,"ATX1"
380 LINE INPUT #1,K$: PRINT K$;
390 IF K$ <> CHR$(&HA)+"OK" THEN  380
400 FOR N = 1 TO 1000: NEXT N
410 CLS
420 INPUT "Auto Dial/Auto Answer ? ( D,A ) ",AU$
430 IF AU$ = "A" OR AU$ = "a" THEN 1160
440 '*********************************************
450 '*              AUTO DIAL                    *
460 '*********************************************
470 PRINT #1,"ATS0=0"
480 LINE INPUT #1,K$ : PRINT K$;
490 IF K$ <> CHR$(&HA)+"OK" THEN 480
500 FOR N = 1 TO 1000: NEXT N
510 FLAG$="D"
520 CLS
530 PRINT "AUTO DIAL MODE"
540 '*********************************************
550 '*           INPUT DIAL NUMBER               *
560 '*********************************************
570 PRINT "Dial number ? ";
580 J = 0
590 D$(J) = INKEY$
600 IF D$(J) = "" THEN 590
610 IF D$(J) = CHR$(8) THEN B$ = CHR$(&H1D)+" "+CHR$(&H1D)
    :PRINT B$;:J = J-1 :GOTO 590
620 PRINT D$(J);
630 IF D$(J) = CHR$(&HD) THEN 670
640 J = J+1
650 GOTO 590
660 DIALNO$(0) = ""
670 FOR L = 0 TO J
680 DIALNO$(L+1) = DIALNO$(L)+D$(L)
690 NEXT L
700 PRINT #1,"ATD"+DIALNO$(J+1)
710 '*********************************************
720 '*         WAIT CONNECT MESSAGE              *
730 '*********************************************
740 K$ = INKEY$
750 IF K$ <> "" THEN  PRINT #1,K$;
760 IF LOC(1) = 0 THEN 740
770 LINE INPUT #1,K$:PRINT K$
780 IF K$ = CHR$(&HA)+"CONNECT" THEN 850
790 IF K$ = CHR$(&HA)+"CONNECT 1200" THEN 970
800 IF K$ = CHR$(&HA)+"NO CARRIER" THEN 1090
810 GOTO 740
820 '*********************************************
830 '*              CONNECT 300                  *
840 '*********************************************
850 N = INP(&H3FB)
860 N = N OR &H80
870 OUT &H3FB, N
880 OUT &H3F8,&H80
890 OUT &H3F9,&H1
900 N = INP(&H3FB)
910 N = N AND &H7F
920 OUT &H3FB,N
930 GOTO 1340
940 '*********************************************
950 '*             CONNECT 1200                  *
960 '*********************************************
970 N = INP(&H3FB)
980 N = N OR &H80
990 OUT &H3FB,N
1000 OUT &H3F8,&H60
1010 OUT &H3F9, &H0
1020 N = INP(&H3FB)
1030 N = N AND &H7F
1040 OUT &H3FB,N
1050 GOTO 1340
1060 '*********************************************
1070 '*               RE-DIAL                     *
1080 '*********************************************
1090 IF FLAG$ = "A" OR FLAG$ = "T" THEN 1340
1100 INPUT "Re-Dial ? (Y,N) ",RD$
1110 IF RD$ = "Y" OR RD$ = "y" OR RD$ = "" THEN 700
1120 GOTO 1340
```

```
1130 '*********************************************
1140 '*                 AUTO ANSWER               *
1150 '*********************************************
1160 PRINT #1,"ATS0=1"
1170 LINE INPUT #1,K$:PRINT K$;
1180 IF K$ <> CHR$(&HA)+"OK" THEN 1170
1190 FOR N = 1 TO 1000:NEXT N
1200 CLS
1210 PRINT "AUTO ANSWER MODE "
1220 FLAG$ = "A"
1230 B$ = INPUT$(LOC(1),#1)
1240 K$ = INKEY$
1250 IF K$ = CHR$(&H1B) THEN 1340
1260 IF LOC(1) = 0 THEN 1240
1270 LINE INPUT #1,K$:PRINT K$;
1280 IF K$ <> CHR$(10) + "RING" THEN 1270
1290 K$ = INPUT$(LOC(1),#1)
1300 GOTO 740
1310 '*********************************************
1320 '*             TERMINAL  MODE                *
1330 '*********************************************
1340 CLS
1350 PRINT CHR$(13)+CHR$(10)+"TERMINAL MODE"
1360 FLAG$ = "T"
1370 KEY ON
1380 COM(1) ON
1390 KEY(1) ON
1400 KEY(2) ON
1410 KEY(3) ON
1420 KEY(4) ON
1430 KEY(10) ON
1440 IF FLAG$ = "S" THEN GOSUB 2080:GOTO 1440
1450 K$ = INKEY$
1460 IF K$ = CHR$(&H1B) THEN 1490
1470 IF K$ <> "" THEN PRINT #1,K$;
1480 GOTO 1440
1490 GOTO 1560
1500 KEY OFF
1510 CLS
1520 GOTO 280
1530 '************************************************
1540 '*            TERMINAL OPERATION                *
1550 '************************************************
1560 COM(1) STOP
1570 KEY(1) STOP:KEY(2) STOP:KEY(3) STOP:KEY(4) STOP
     :KEY(10) STOP
1580 FOR N =1 TO 2000:NEXT N
1590 PRINT #1,"+";
1600 FOR N = 1 TO 50:NEXT N
1610 PRINT #1,"+";
1620 FOR N = 1 TO 50 : NEXT N
1630 PRINT #1,"+";
1640 FOR N = 1 TO 2000:NEXT N
1650 PRINT #1,"ATZ"
1660 FOR I = 1 TO 1000:NEXT
1670 K$ = INPUT$(LOC(1),#1)
1680 RETURN
1690 '***********************************************
1700 '*           ON COM OPERATION (RECV)           *
1710 '***********************************************
1720 IF LOC(1)>128 THEN PAUSE = TRUE :PRINT #1,XOFF$;
1730 FOR I = 1 TO LOC(1)
1740 B$ = INPUT$(1,#1)
1750 IF FLAG$= "R" THEN PRINT #3,B$;:GOTO 1800
1760 IF B$ = CHR$(8) THEN B$ = CHR$(&H1D)
1770 IF B$ = CHR$(10) THEN B$ = ""
1780 IF B$ = CHR$(19) THEN XFLAG$ = "XOFF"
1790 IF B$ = CHR$(17) THEN XFLAG$ = "XON"
1800 PRINT B$;
1810 NEXT I
1820 IF LOC(1)<128 AND PAUSE = TRUE THEN PAUSE = FAULT
     : PRINT #1, XON$;
1830 RETURN
1840 '***********************************************
1850 '*         F1 ON  KEY (FILE SEND)              *
1860 '***********************************************
1870 PRINT CHR$(13) + CHR$(10) + "SEND FILE MODE"
1880 KEY(1) STOP: KEY(2) STOP
1890 INPUT "Send file name ? ",SN$
1900 OPEN SN$ FOR INPUT AS #2
1910 FLAG$ = "S"
1920 XFLAG$ = "XON"
1930 PRINT CHR$(10)+"SENDING "+SN$+" ... "
1940 RETURN
1950 '***********************************************
1960 '*          F2 ON KEY (FILE RECV)              *
1970 '***********************************************
1980 PRINT CHR$(13)+CHR$(10)+"RECEIVE FILE MODE"
1990 K$ = INPUT$(LOC(1),#1)
2000 KEY(1) STOP: KEY(2) STOP
2010 INPUT "Receive file name ? ",RN$
2020 OPEN RN$ FOR OUTPUT AS #3
2030 FLAG$ = "R"
2040 RETURN
2050 '*********************************************
2060 '*        FILE SEND SUBROUTINE               *
2070 '*********************************************
2080 IF XFLAG$ = "XOFF" THEN RETURN
2090 IF EOF(2) THEN GOTO 2130
2100 K$ = INPUT$(1,#2)
2110 PRINT #1,K$;
2120 RETURN
2130 PRINT CHR$(13)+CHR$(10)+"FILE SEND COMPLEX !"
2140 FLAG$ = "T"
2150 '*********************************************
2160 '*          F3 ON KEY (SEND END)             *
```

```
2170 '********************************************
2180 CLOSE #2
2190 FLAG$ = "T"
2200 RETURN 1350
2210 '********************************************
2220 '*            F4 ON KEY (RECV END)           *
2230 '********************************************
2240 CLOSE #3
2250 FLAG$ = "T"
2260 RETURN 1350
2270 '********************************************
2280 '*              PROGRAM END                  *
2290 '********************************************
2300 GOSUB 1560
2310 CLOSE #1:CLOSE #2:CLOSE #3
2320 CLS
2330 END
```

성심치과의원

취재/이향선 기자

최첨단 기술로 손꼽히는 컴퓨터가 대량으로 보급되면서 이제 우리는 사회의 구석구석에서 컴퓨터의 이용이 실용화되고 있다는 것을 피부로 느끼고 있다.

이러한 최첨단의 컴퓨터가 최첨단의 의술인 치과의료계에 파문을 일으키고 있는 그 현장을 찾아가 보았다.

치과의사의 호기심에서 시작

압구정동의 성심치과(원장:박은기)를 들어서면 아담한 병원분위기와 상냥한 간호원의 미소를 압도하는 컴퓨터와 프린터를 만나게 된다. 그리고 원장실로 들어서면 의연한 16비트 컴퓨터와 아주 오래된 보물마냥 소중히 싸여진 애플컴퓨터와 여러종류의 프린터 용지를 보게 된다.

이곳은 치과의사의 방이 아니라 차라리 컴퓨터광의 방이라 해야 더 옳을 듯하다.

박은기원장이 컴퓨터를 사용하게 된 것은 치과의사로서의 절실한 필요성에 의해서라기 보다는 다분히 일반적이고 단순한 호기심 때문이었다. 그는 82년 병원을 개원하면서 8비트 컴퓨터를 구입하여 처음 컴퓨터와 만나게 된다. 그 후 컴퓨터에 대하여 전문가들에게 물어 물어 어느 정도 익숙하게 되자 자신의 분야에도 컴퓨터를 이용해야겠다는 욕심(?)이 생기게 되었고 87년 2월에는 16비트 컴퓨터 세트를 완전히 구비하였다.

의료보험청구에 컴퓨터를 처음 이용

박원장이 처음으로 치과원무에 컴퓨터를 이용한 것은 83년 12월 8비트 퍼스널컴퓨터를 이용한 의료보험청구 업무에서 였다. 의료보험청구는 병원에서 가장 번거로운 일이기 때문에 이 부분이 가장 먼저 전산화가 필요하였던 것이다. 의료보험청구에 대한 컴퓨터의 이용이 어느 정도 치과의사의 고민을 해결하게 되자 환자관리, 통일된 처치코드를 이용한 관리시스템도 개발하여 차츰 컴퓨터의 이용도가 넓어지게 되었다.

그러나 8비트 컴퓨터를 이용하였을 경우 종합적인 원무의 처리가 어렵다는 문제성이 대두되자 방안을 모색하던중 16비트에서 그 처리가 신속하고 정확하게 이루어지는 치과관리용 프로그램인 덴컴-II가 87년 11월에 개발되어 치과의원의 복잡한 업무를 능률적이고 효과적인 업무로 대치시키게 되자 박원장도 이를 자신의 병원 업무에 도입하여 현재 이를 이용하고 있다.

치과관리용 프로그램 덴컴-II

앞에서 설명한 바와 같이 8비트에서는 부분적인 처리만 하게 되자 이를 통합한 원무절차가 필요하게 되었고, 치과원무의 까다로운 절차를 16비트에서 보완한 덴컴-II는 편리한 기능과 함께 번거로운 잡무를 처리하여 치과의사에게 더 향상된 진료와 시간의 여유를 선사하게 된다.

이 덴컴-II의 특징은 각 환자의 치료 후 즉시 또는 하루의 진료가 끝난 후 치료내역이나 인적사항을 컴퓨터에 입력하면 입력 즉시 각종 자료들을 출력하거나 통계의 산출이 이루어지도록 구성되어 있다.

또한 덴컴-II는 컴퓨터를 접수대에 설치하고 초진환자의 등록, 치료사항의 등록, 치료사항의 입력, 다음 내원시의 배정, 지출내역의 입력 등을 처리하고, 마감시에 거래처결재 사항 등을 입력하여 일일정산(환자집계, 수지집계)을 출력하며, 월말에는 월정산을 출력하거나 보험청구를 하는 것이다.

이러한 과정에서 수시로 RECALL CHECK나 특별관리가 필요한 환자를 추출하여 전화, 우편, 방문이나 다른 방법으로 환자의 병원에 대한 인식을 새롭게 할 수 있고 의사에 대한 신뢰감을 심어 주도록 하였다. 그리고 이러한 기능들을 사용자가 자신에게 맞는 방법을 선택하여 진료에 이용할 수 있게 약간의 사용규칙만 바꾸어 다양하게 사용하도록 하였다.

하드웨어와 OS

덴컴-II를 이용하려면 IBM PC XT/AT와의 호환성을 갖는 16비트의 컴퓨터로 최소한 10MB 이상의 하드디스크와 한개 이상의 플로피디스크 드라이브, 한행에 80자(1:2 모드에서는 96자) 이상 인쇄할 수 있는 프린터가 필요

하다.

사용할 수 있는 OS로는 버전 3.1 이상의 MS-DOS이며 사용자는 DIR, COPY, DATE, TIME, DEL, RENAME, TYPE, BACKUP, RESTORE, MD, CD, RD, FORMAT, PATH 등의 간단한 DOS 명령어를 사용할 수 있어야 한다. 또한 프린터와 호환성있는 한글과 메일 머지 기능을 가진 워드프로세서를 어느 정도 활용할 수 있어야 한다.

또 덴컴-II는 RM 코볼 버전 2.0으로 작성되었으며 차트의 관리기능, 스케줄 관리기능, 원무 관리기능, 의료보험 청구기능, 각종 통계자료의 추출 및 우편물의 발송기능, 각종 코드 관리기능, 수가 변경시 변경전 데이터 별도 보관 기능으로 구성되어 있고 메인메뉴 화면은 화면 1과 같다.

화면 1에 나타난 기능중 중요한 몇가지의 기능은 다음과 같다.

●차트 관리기능

차트는 Patient Record, Dental Record, Treatment Record 등 세가지 코드로 구성되는데 Patient Record는 환자의 인적사항을 입력하여 각종 통계자료를 추출하는 기본사항들이 포함되고 모든 항목의 입력시나 수정시 최다 확률의 초기값을 표시해 두어 입력이 빠르고 쉬우며 항목간의 이동이 용이하다. 특히 항목의 이름과 상관없이 개인적인 신상자료철로도 사용할 수 있다.

Dental Record는 치아별로 진찰상태, 진료후의 진행상태 또는 재발상태를 입력할 수 있으며, 전신 및 구강상태의 많은 항목이 간편한 입력으로 처리되어 치료계획을 세우기 쉽도록 구성되어 있다.

Treatment Record는 환자별로 진료기록을 입력하는 레코드로 지나간 진료기록을 보면서 새로운 치료 사항을 기록할 수 있고 무순으로 입력해도 날짜순으로 배열되어지며 보험환자의 일반진료도 가능하다.

●스케줄 관리기능

스케줄 관리는 일자와 시간별로 약속을 입력하거나 확인하는 '약속표'와 각 주일별 약속사항을 입력하거나 확인하는 '주일별 약속현황'으로 구성되어 있다.

●원무 관리기능

치과에서의 원무관리는 의사의 시간적 제약이나 환자진료에 과다한 노력이 소모되는 부분인데 이 프로그램은 일일 환자집계표, 월 환자집계표, 지출명세서, 제약회사 및 재료상 관리로 각사의 구입내역, 결재금의 사항을 입력하여 현재의 잔고를 항상 확인할 수 있고 이월작업이 가능하도록 한 거래처관리A, 기공소의 거래나 개인적인 어느 거래의 기록도 입력할 수 있게 한 거래처관리B, 약품 및 재고관리에 대한 기능을 수행한다.

●의료보험 청구업무

의료보험은 우선 청구처리속도의 신속과 정확성에 그 주안점을 두고 진료비명세서를 출력하기전에 월별, 보험종류별로 Patient Record와 Treatement Record의 내용을 토대로 정액, 정률, 2,000원 미만 등으로 분류하는 의료보험 출력전처리, '출력전 처리된 데이터를 토대로 각 개인별 총 진료비, 청구액은 정률과 정액, 정액순으로 번호를 부여하여 출력하는 총진료비, 일람표 출력, 진료비 명세서의 일부나 전체의 출력, 단일 출력의 기능을 수행한다.

●각종 통계자료 추출 및 우편물 발송기능

치과관리 업무중의 중요한 부분은 원하는 통계의 신속한 확인과 그 자료를 이용하여 환자관리에 이용하는 것이다.

이 기능은 약속 일자별 리스트, 생일별 환자 리스트, 소개자 리스트, 미수금 리스트, 직업별 환자 리스트 등의 통계 기능과 이들에 의해서 추출된 데이터를 이용하여 우편물의 자동인쇄 기능을 실현할 수 있게 하였다.

●각종 코드 관리기능

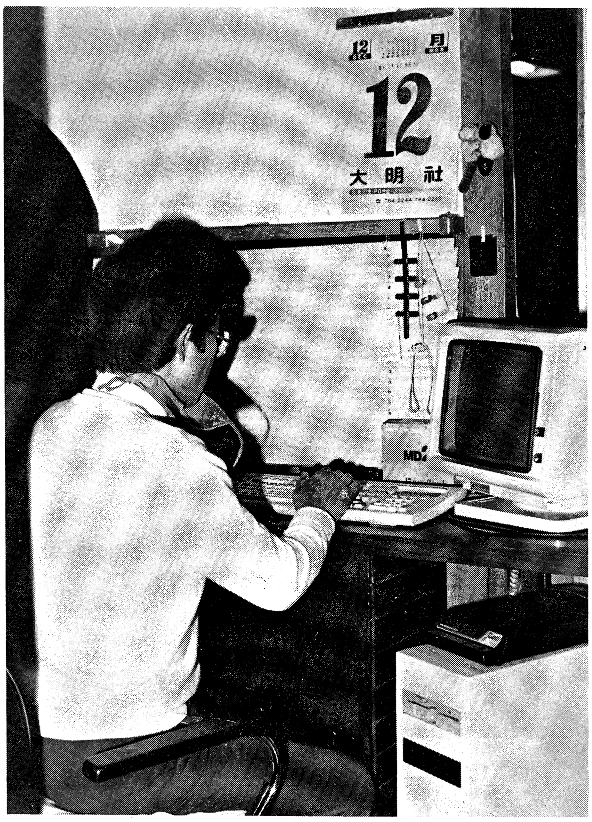
이 기능은 입력방법이나 수정, 삭제가 쉽고 화면에 순서적으로 표시된 상태를 항상 확인하며 입력과 수정이 이루어져 인쇄된 코드표를 일일이 찾을 필요가 없다.

〔화면 1〕 덴컴-II의 메인메뉴 화면

```
DENCOM-II
Current Date:800101                                  Valid user :  성 심 치과의원

        MAIN MENU

    1.  Patient   Record
    2.  Dental    Record
    3.  Treatment Record
    4.  Schedule                           DENTAL
    5.  Management & Report                COMPUTERIZED PROGRAM
    6.  List & Mail-merge                  OF ALL THE CLINIC
    7.  Insurance    Print                 MANAGEMENT PACKAGES.
    8.  Sub MENU
    9.  Old times    Report
    0.  Operating-MENU                 System analysed by Dong han, Lee
                                                   D.D.S.,M.S.D.,
Press [CapsLock] & [NumLck],   Move for [Up] & [Down],   Select with [Return]
COPYRIGHT 1987. PIONEER COMPUTER.
```



단순한 호기심으로 컴퓨터를 사용하였지만 제2의 진료도구로 컴퓨터를 이용할 것이라고.

●수가 변경시 변경전 데이터 별도보관 기능

의료보험의 진료 수가의 변동이 발생하면 미 청구된 진료비의 청구를 완전히 끝낸 후 진료수가 기준액표를 수정해야 하는데 수정 전의 진료기록은 그 후에 수정하면 수정된 수가로 재계산되어 문제가 발생하므로 그것을 플로피디스크로 보관하였다가 수정 청구나 재청구에 이용하도록 한 기능이다.

보급현황과 문제점

현재 전국적으로 치과의사들이 보유하고 있는 시스템은 8비트가 100여대, 16비트가 200여대 정도 되며 주로 의료보험청구와 같은 자료 및 통계를 위해 사용하고 있고, 덴컴-II를 이용하고 있는 의원수는 전국적으로 100개 의원 정도가 된다. 또한 앞으로 더많은 치과의사들이 이 프로그램을 이용할 것으로 예상되고 있다.

그러나 이 덴컴-II도 완벽한 것은 아니며 코볼이 그래픽처리가 어렵다는 점을 가지고 있기 때문에 그에 대한 대책과 현재의 복잡한 기능을 단순화해야 한다는 면과 일정한 나이 범위에서 생일별로 추출할 수 있는 기능 등 검색 조건을 다양화시켜 원하는 자료를 간단한 조작으로 검색할 수 있게 하는 방법, 그리고 코볼이 다른 패키지보다 속도면에서 떨어지므로 속도를 개선해야 하는 점이 보완되어야 할 것이다.

그러나 가장 큰 문제점으로 박원장은 아직도 많은 치과의사의 컴퓨터에 대한 인식이 그리 확산되지 않고 있는 실정이라는 점을 든다. 그리고 단순히 관리프로그램을 사용하는데 치과의 전산화가 있는 것이 아니기 때문에 원무에 사용되는 것 뿐만 아니라 컴퓨터가 치과의료의 다른 분야에도 이용되어야 한다고 보고 있다.

예를 들면 교정할 때 X-ray를 찍어 수치를 계산하고 그림을 그려 처리하는 등 매우 복잡한 절차에 대한 문제점을 간단하게 처리할 수 있는 교정진단 분석프로그램이나 컴퓨터와의 대화(여기서 컴퓨터와 대화한다는 것은 의사가 환자와 직접 대화하는 것이 아니라 질문을 주고 환자가 의사와 직접 대화하면서 주관적으로 표현되는 반응의 차이를 줄이기 위해 컴퓨터에 작성하도록 하여 좀더 객관적인 자료를 통하여 진료하기 위한 방법)를 통하여 그 결과를 분석하여 치료에 적응하는 분석프로그램이 시급하다는 것이다.

그러므로 이러한 문제점을 극복하려면 치과계 자체의 활발한 활동이 있어야 하며 특히 컴퓨터 전문가와 치과의사가 서로 연계성을 가지고 양자의 발전을 위한 노력을 경주해야 할 것이다.

컴퓨터와 치과의사의 만남이 결코 생소한 것이 아닌 지금의 시대에 고급의 치과문화(치과의사의 입장에서 새로운 문화를 유도하는 의미에서 비롯된 단어) 보급과 컴퓨터 마인드확산이라는 과제를 두고 치과의사나 컴퓨터 전문가가 해결해야 할 일은 산재되어 있다. 이 시점에서 컴퓨터를 이용하는 새로운 시도가 치과계 및 타 의료계로 확산시키려는 의지가 결집되어 의사와 컴퓨터의 불가분의 관계가 빠른 시일에 이루어지길 기대해 본다.

스스로 손쉽게 메모리를 증설한다.

메모리 확장

수메가바이트 메모리를 갖고 있는 친구들의 컴퓨터와 자신의 컴퓨터를 비교한다면 자신의 컴퓨터가 보잘것 없는 것으로 보일지도 모른다.

아마 여러분은 메모리 확장 보드나 다기능 카드(멀티펑션카드)를 보고난 후 자신의 메모리 소켓이 비어있다면 빈 소켓을 채우기 위해 RAM을 구입하러 용산이나 청계천등의 상가로 달려갈지도 모른다. 일단 메모리가 클 경우 여러분은 보다 큰 응용프로그램을 수행시킬 수 있을 것이고, RAM디스크나 스풀러로 이용하기도 하고 주기억장치의 용량에 제약을 주는 유틸리티들도 이용할 수 있을 것이다.

이 글에서는 이용자가 필요로 하는 칩의 용량이 64K비트가 좋을지 256K비트가 좋을지 256K비트 칩을 구한다면 DIP(Dual-In-line Package)을 선택할지 SIM(Single In-line Module) 칩을 선택할지도 인도해 줄 것이다.

●칩의 형태는

여러분의 컴퓨터가 빈 RAM소켓을 갖고 있으면 RAM칩을 마더보드, 다기능 카드, 메모리 확장카드 중 어느 곳에라도 설치할 수 있다. 빈 메모리 보드를 0K 또는 비거주(Un-populated) 확장보드라고도 부른다. RAM은 보통 컴퓨터 BIOS를 저장하는 ROM(Read Only Memory)과는 달리 읽기와 쓰기가 다 가능한 기억소자이다.

또 RAM은 근본적으로 정적인(Static) RAM과 동적인(Dynamic) RAM의 2가지가 있다. 정적인 RAM에서 각 메모리의 위치는 작은 전기 스위치와 같아서 한번"on"이나"off"로 정해지면 컴퓨터가 그것을 변화시키기까지 고정된다. 한편, 동적인 RAM은 작은 축전기(Capacitor)의 모임이다. 그래서 정보를 저장하기 위해서는 충전과 방전의 작용이 있게 된다. 충전은 빨리 사라지므로 4ms마다"refresh"시켜야 한다. 이와같이 정적인 RAM의 선택이 논리적인 것처럼 보이나 가격면에서 동적인 RAM보다 비싸다. 그래서 대부분 MS-DOS를 이용하는 컴퓨터에서는 동적인 RAM을 사용한다.

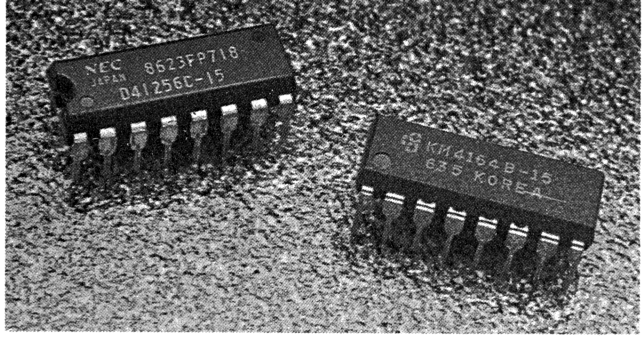
전형적인 256K비트 RAM 칩에서는 매 4ms마다 256번의 "refresh"가 필요하다. 즉 15.6μs마다 컴퓨터가"refresh"를 위해 멈추게 되어 전체 컴퓨터 시간의 7%정도가 동적인 RAM을 위해 소요된다. 대부분의 컴퓨터와 메모리보드는 64k비트와 256K비트 동적인 RAM(DRAM) 칩을 이용한다. 칩을 선택하기 위해 컴퓨터 내부를 보면 이미 칩들이 있을 것이다. 64K비트 칩의 경우 4164이고 256K비트 DRAM은 41256으로 칩에 써있다. 그러나, 히다찌 256K비트 DRAM은 50256으로 써있다.

●메모리 확장은 어떻게

각 메모리 주소는 8비트 길이가 1바이트를 이루므로 8개의 칩이 모여서 1바이트를 구성하게 된다. 다시 말해서 64K비트 칩들은 64K바이트 메모리를 구성하며 8개의 256K비트 칩이 모여 256K바이트의 메모리를 만들게 되는 것이다. 그러나 실제로는 9개의 칩이 각 바이트를 구성하게 되는데 그것은 패리티를 위해 쓰인다.

메모리에 쓰기를 할때마다 하드웨어가 1로 지정된 바이트의 비트수를 헤아려 홀수패리티 또는 짝수패리티 검사방식에 근거하여 패리티 칩에 기록하게 된다. 나중에 다시 읽기 수행을 할 경우 하드웨어가 지정된 비트를 검사하여 이상이 있을 경우 메모리 결함이 있음을 컴퓨터에게 알리고 컴퓨터는 다시 부트(Reboot)할 때까지 멈춰있게 된다. 그러므로, 64K바이트나 256K바이트 뱅크마다 9개의 칩을 구해야 할

64k비트와 256k비트 DRAM칩



것이다. 그러나 메모리카드 한행의 빈소켓 수를 세어 8개의 칩이 필요할지 9개의 칩이 필요할지 알아야 한다. 또 이런 지침을 따르지 않는 컴퓨터도 있다. 512K 바이트 용량의 컴퓨터를 640K 바이트로 늘리기 위해서는 4비트×64K짜리 4개가 필요하다. 이런 칩은 각 바이트 당 4비트씩을 구성하고 있다.

외부에 코드번호가 찍힌 경우를 제외하고 64K 비트와 256K 비트 칩은 보통 동일하게 보인다. 보통 둘다 16핀 DIP 칩으로 제공된다. 1M 비트 칩은 각각 9개씩의 핀을 가져서 총 18개의 핀을 갖고 있다. 보통 핀의 수가 그 칩의 크기를 나타내는데 핀을 추가함으로써 낭비되는 컴퓨터 공간을 줄이기 위해 요즘의 몇몇 메모리카드들은 하나의 SIM 패키지에 여러 256K 비트 칩들을 사용한다.

SIM은 작은 PCB(Printed Circuit Board) 조작이라 말할 수 있다. 전체 모듈을 메모리카드에 있는 코넥터에 연결된다. SIM 패키지에 있는 256K 비트 칩들은 고전적인 DIP 칩들보다 한비트가 많다. 그러나 제한된 공간에 보다 많은 메모리를 확보할 수 있는 장점을 지니고 있다.

●액세스 속도는

칩을 살때 고려해야할 또 하나의 사항이 속도이다. 컴퓨터가 메모리를 액세스하려고 할때 번지를 전기적으로 생성해 낸다. DRAM 칩들은 해당 번지의 각부분을 얼마나 빨리 읽어들이는가 정확한 정보를 갖고 있는 비트를 발견하여 CPU에 그 비트를 얼마나 빨리 다시 보내는가에 의해 평가된다.

이때 필요로 하는 전체시간을 칩의 액세스 속도라 하며 ns(nanoseconds)나 그이상의 단위로 측정된다. 따라서 칩 구입시는 소유하고 있는 컴퓨터를 유지하는데 충분한 속도를 갖고 있는 칩을 구입해야 할 것이다. 그렇다고 무조건 빠른 것만 구한다고 장점이 있는 것은 아니다.

액세스 속도가 200ns 정도의 칩들은 4.77MHz로 동작하며 CPU가 8088인 XT 호환 기종에 적합하다. 그러나, 150ns, 120ns 등 그보다 빠른 경우도 괜찮다. 액세스 속도는 보통 부품번호와 함께 칩상단에 찍혀있는데 마지막 자리는 항상 0이며 생략된다. 예를 들어 150ns, 256K 비트 칩은 41256－15라고 써 있다. 만일 메모리보드에 이미 여러 칩이 설치되어 있는 경우 그와 동일한 부품번호와 액세스 속도를 가진 칩을 구하는 것이 좋을 것이다.

메모리보드에 DRAM을 사용할 경우는 반도체 제조회사의 매뉴얼에 나와 있는 액세스속도를 참조하면 도움이 될 것이다. 그것들이 값이 싸다면 항상 보다 빠른 칩을 선택하되 빠른 액세스 속도가 반드시 여러분의 컴퓨터 수행속도를 증가시켜주지는 않는다는 사실을 알아야 한다.

●설치순서는

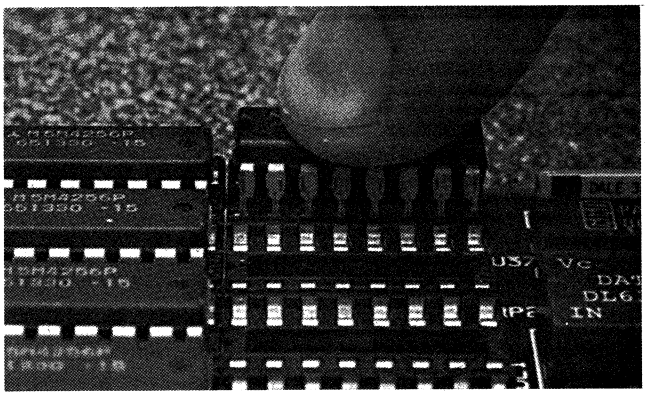
아직도 어떤 종류의 칩을 선택할 지 혼동된다면 메모리보드나 컴퓨터와 같이 딸려오는 매뉴얼 등을 읽어보면 도움이 될 것이다. 만일 그런 책자가 없다면 구입한 회사에 전화를 걸어 메모리에 관한 문의를 하여도 된다.

메모리를 추가할때 컴퓨터나 메모리카드 내부에 있는 몇몇 스위치를 조정해야 하는 것이 필요하다. 이때 컴퓨터나 메모리 카드에 관한 책자에는 이 DIP 스위치에 대한 설명이 있을 것이다. 몇몇 책자에는 마더보드로 연결된 점퍼 선등을 제거하는 것을 설명하는 것도 있다. IBM 레퍼런스 매뉴얼을 보면 딥스위치 세팅에 대해 자세히 나와 있다.

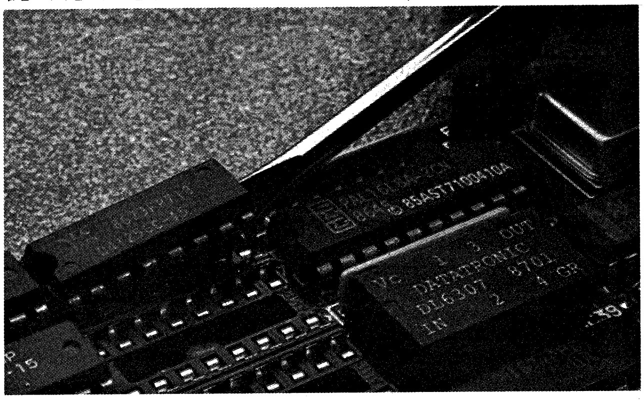
컴퓨터 내의 다른 소자들과 마찬가지로 DRAM도 정적인 RAM보다 민감하다. 그래서 작은 정전기 쇼크도 그 칩을 못쓰게 만들 수 있으므로 조심해야 한다.

설치하는 첫번째 단계가 작업해야 할 공간을 찾는 것이다. 가능하면 나무나 타일로 된 마루를 가진 방에 큰 테이블이나 작업대를 선택하는 것이 좋다. 작업하는 동안은 개나 고양이, 카페트 같은 요소는 피해야 한다. 카페트가 있는 방에

DRAM칩을 삽입할 때는 너무 무리한 힘을 가하지 않고 조심스레 손가락으로 누른다.



칩을 제거할 때는 일자드라이버 등으로 조금씩 힘을 주어 구부러지지 않도록 조심한다.



서 작업할 때는 되도록이면 고무나 합성물질을 발바닥에 둔다. 플러그를 컴퓨터에서 제거하고 모니터, 키보드 등 다른 외부장치를 제거한다.

컴퓨터를 작업대에 놓고 몸에 있을지도 모르는 정전기를 제거하기 위해 주위에 금속으로 된 부분에 손을 댄다. 다음 컴퓨터의 덮개를 제거한다. 컴퓨터 안쪽 뒷구석 오른쪽으로 경고표시가 있는 금속상자를 보게 될 것이다. 이것이 컴퓨터의 전력 공급장치이다. 여러분과 컴퓨터가 같은 접지 전압을 갖기 위해 다른 것에 손을 대기전에 이것을 누른다.

이전에 메모리를 추가한 적이 있는 메모리보드에 칩을 추가하려면 그 보드를 먼저 제거한다. 이 메모리카드는 금속받침과 스크류(Screw)를 갖고 있으며, 컴퓨터의 뒤에 붙어 있을 것이다. 그 스크류를 조심스럽게 제거한다. 여러분이 재수가 없으면 그 스크류가 손에서 튀어나가서 컴퓨터내에 떨어져 마더보드아래 숨어 버릴 수도 있다. 그래서 조심스럽게 다루어야 한다. 이제 메모리카드를 똑바로 위로 당긴다. 앞뒤로 흔들면서 천천히 빼낸다. 그 카드를 테이블에 놓는다. 만일 마더보드에 칩을 설치한다면 메모리카드를 제거하는 일은 필요치 않는 점을 주의하기 바란다. 그러나 원하는 메모리 소켓에 칩을 설치하기 위해서는 다른 카드를 빼야 될지도 모른다.

이제 사들고온 칩을 알루미늄박 위에 놓는다. 칩의 한쪽 끝을 보면 표시가 되있거나 점이 찍혀있거나, 작은 동그라미나 파낸 자리가 있음을 주지하기 바란다. 한쪽 끝에만 표시된 부분을 찾는다. 이 표시는 소켓에 끼워 넣을 때 그 칩이 어느 방향으로 놓일지 지시한다. 표시된 부분을 위로 놓고 볼때 표시된 부분의 왼쪽 끝이 1번 핀이다.

칩을 어떻게 꽂을지 확인을 해둔다. 칩의 핀이 조금 벌어져서 소켓의 구멍에 맞지 않으면 손으로 조금 구부린다. 칩을 꽂을 경우 너무 힘있게 누를 필요는 없다. 핀이 이제 고르게 펴지면 소켓에 핀을 놓고 대응하는 구멍에 맞춘다. 이제 단단히 아래로 누른다. 소켓에 완전히 자리 잡을 때 까지 눌러준다. 칩의 핀이 구부려진 곳이 있나 주의깊게 살핀다. 다른 칩들도 꽂는다. 설명하는 것보다 실제 작업은 훨씬 쉽다. 올바르게 했다고 생각되면 다른 빈곳도 채운다. 이때 점퍼연결을 제거한다든지 메모리확장에 따른 스위치 조작이 있음을 기억하기 바란다.

모든 작업이 끝나면 컴퓨터를 재조립 한다.

●메모리 검사는

거의 모든 MS-DOS 컴퓨터는 POST(Power On Self Test)를 하는 동안 메모리 검사를 한다. 이 검사를 잘 통과하면 모든 것이 잘 끝난 것이다. 컴퓨터가 메모리 오류를 통보하면 모든 플러그를 제거하고 컴퓨터를 분해해서 작업한 것을 조심스레 검사해 본다. 보통 칩의 핀이 구부러진 것이 많이 발견되므로 이것부터 찾아본다. 칩을 뺄 경우는 일자드라이브 같은 것으로 한쪽 쪽을 조심스럽게 들어 올린다.

일단 POST를 통과하면 MS-DOS의 CHKDSK 프로그램을 수행하여 새로 추가한 메모리를 확인하도록 한다. 여기에서 이상이 있으면 메모리 카드나 컴퓨터 내부의 딥 스위치들 중 어느 한 곳이 잘못 조정되어 있는지 확인한다.

자세히 체크하기를 원하면 자기 진단 프로그램 등을 구해 수행시켜 보는 것도 좋은 방법이다. 경험이 풍부한 기술자는 이런 모든 일련의 과정을 10분이내에 해치울 것이다. 경험이 없어도 20~30분이면 될 것이다. 자신의 기술이 미심쩍은 사람은 주위의 경험자에게 물어보면 잘 알려줄 것이다. PC

애플 데스크탑 버스

매킨토시 SE내의 ADB 트랜시버 칩(Transceiver Chip)은 온 보드 TTL 신호와 ADB에 사용되는 가변펄스폭 개방콜렉터 신호사이를 변환하고 버스가 쓰이고 있지 않을때 최종적으로 데이터를 보낸 ADB 디바이스의 자동 폴링(Polling)을 수행한다.

매킨토시 SE는 TTL 신호를 보내고 받기위해 VIA내의 시프트 레지스터를 사용하여 자기자신의 ADB 트랜시버와 데이터를 주고 받는다. 시프트 레지스터는 ADB 트랜시버로부터 안으로 시프트되는 8비트나 또는 밖으로 시프트되는 8비트를 포함한다.

ADB 데이터 선들 이외에도 VIA와 ADB 트랜시버사이에 여러개의 제어 선들이 있다. 이러한 제어 선들은 VIA와 트랜시버 사이의 시간과 시퀀스 처리에 사용된다.

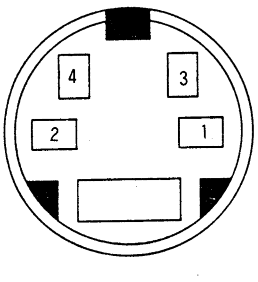
컴퓨터가 처음 부트될 때나 리세트될 때 각각의 ADB 주변장치 디바이스는 $00~$0F사이의 ADB 어드레스로 디폴트된다. 특정 디바이스 형태, 예를들면 상대 위치 디바이스(가령 마우스 디바이스와 같은)에 사용되는 어떤 디폴트 어드레스들은 항상 $03의 디폴트 ADB 어드레스를 갖는다.

각각의 ADB 디바이스는 내부 레지스터내에 저장된 디폴트 인식코드를 갖고 있다. ADB 매니저(Manager)는 디바이스 핸들러(Device Handler) ID의 트랙과 버스상에서 각 디바이스의 디폴트 어드레스를 유지하고 그 디바이스가 컴퓨터에 보낼 데이터를 갖고 있을 때 적당한 디바이스 드라이버를 호출한다.

디바이스는 하나 이상의 기능코드를 갖게 될 수도 있는데, 만일 그렇다면 ADB 매니저는 디바이스 핸들러 ID를 바꾸거나 디바이스로 하여금 기능모드들을 변경시키게 함으로써 디바이스의 내부 레지스터에 라이트 할 수 있다.

ADB 디바이스는 데이터처리를 개시할 수는 없다. 대신에 각각의 디바이스는 컴퓨터에 보낼 데이터를 갖고 있을 때 서비스 리퀘스트 신호를 어서트(Assert) 한다. 그래서 ADB 매니저는 한 디바이스가 데이터를 보내 응답할 때까지 버스상의 각 디바이스들을 리드할려고 시도한다. 한 디바이스가 응답하면, ADB 매니저는 VIA와 ADB 트랜시버를 통해 디바이스와 통신하는 적당한 디바이스 드라이버에 제어를 보낸다.

〔그림 1〕 매킨토시 SE ADB코넥터



〔표 1〕 ADB코넥터 각 신호의 역할

핀번호	신호명	정 의
1	ADB	The bidirectional data bus used for input and output. It is an open-collector signal pulled up to +5 V through a 470 Ω resistor on the Macintosh SE logic board.
2	n.c.	Not connected.
3	+5V	+5V power.
4	GND	The Macintosh SE logic ground.

ADB 인터페이스

ADB 코넥터의 핀 구성은 그림 1 과 같다.

여기서 신호 3은 +5V 전원이며 신호 4는 그라운드에 사용된다. 이들의 핀 번호에 따른 특성은 표 1에서 보는 바와 같다.

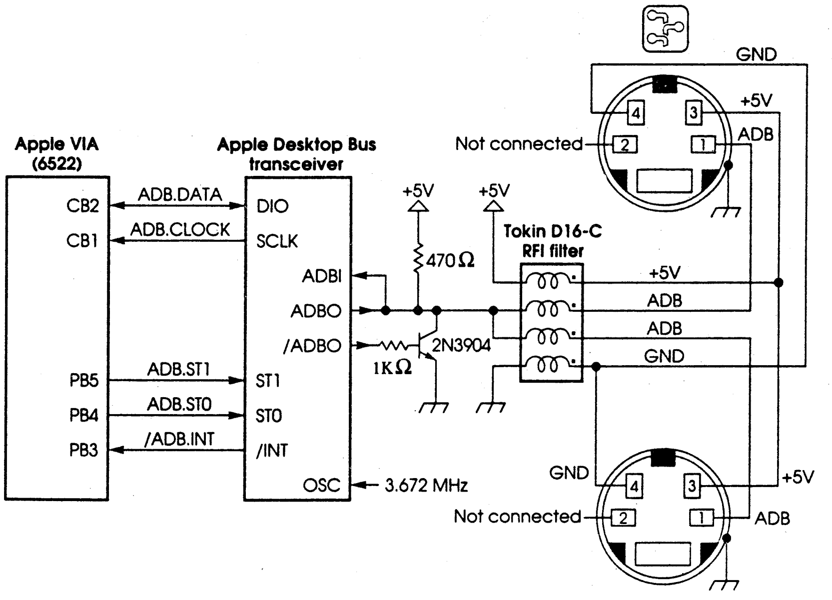
그림 2는 매킨토시 SE ADB 인터페이스에 대한 회로도를 보여준다.

ADB 코넥터의 케이블은 5m이상 길 수 없으며, 케이블, 캐퍼시턴스는 미터 당 100pF 이상되서는 안된다. 비록 ADB는 16개의 다른 주변장치 디바이스까지 어드레스할 수 있는 능력이 있지만, 3개 이상의 디바이스들의 데이지 체이닝(Diaisy Chaining)은 코넥터 저항과 신호 감퇴(Degradation) 때문에 권고되지 않는다.

애플 표준 마우스

애플 표준 마우스는 ADB를 갖고

〔그림 2〕 매킨토시 SE의 ADB 인터페이스 회로



있는 모든 애플 컴퓨터에 사용되고 있다.

애플 표준 마우스내의 마이크로프로세서는 매킨토시 SE에 위치 및 상태 정보를 제공한다. 마우스 메카니즘은 X(왼쪽과 오른쪽)와 Y(위와 아래) 이동을 검지하고 마우스 버튼을 눌렀을 때 그것을 나타낸다. 또한 마이크로프로세서는 ADB 트랜시버 칩으로서 작용하여 컴퓨터내의 ADB 트랜시버 칩과 통신한다. 컴퓨터의 ADB 트랜시버 칩은 마우스 정보를 받아 들여 그것을 VIA에 보낸다.

매킨토시 SE ROM내의 루틴들은 이 정보를 스크린상에서 포인트의 일치하는 이동으로 바꾼다.

마우스는 마우스가 이동함에 따라 마우스 마이크로프로세서에 구형파 신호들을 보냄으로써 동작한다. 마우스내의 볼(Steel ball)은 마우스가 이동함에 따라 회전하며 마우스의 X축 방향으로의 이동은 휠들 중의 하나를 회전시키고 Y축 방향으로의 회전은 다른 휠을 회전시킨다.

라이트의 빔들은 교대로 전송되며 인터럽트 휠내의 슬롯에 의해 차단되며, 광검출기(Proto Detector)는 마우스의 마이크로프로세서에 결과로 구형파 신호를 보낸다. 마이크로프로세서는 이러한 신호들을 인터럽트하여 X와 Y방향으로 이동된 거리를 결정하여 이 정보를 ADB 트랜시버와 VIA를 경유하여 마우스 드라이버 소프트웨어에 보낸다.

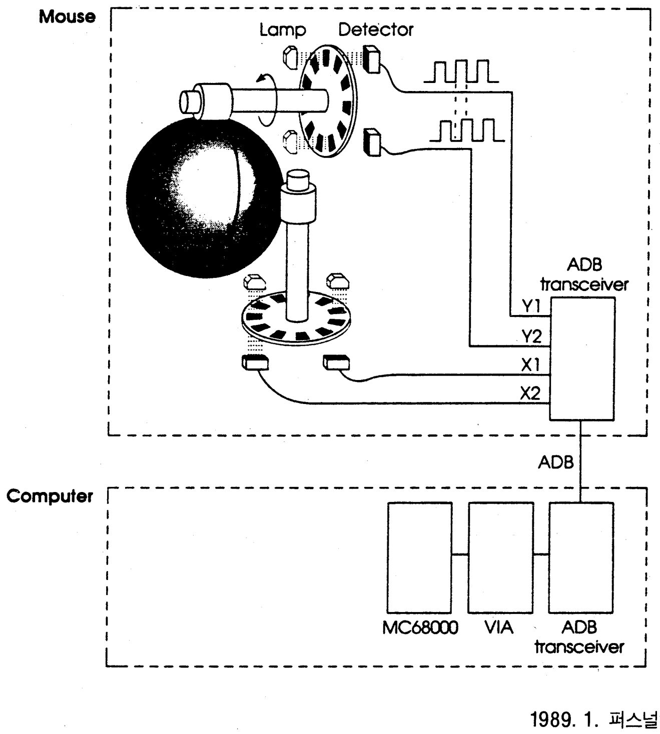
기존의 매킨토시 컴퓨터에 사용된 마우스는 인터럽터 휠에 의해 발생된 구형파 신호들을 컴퓨터에 직접보내며, 여기서 신호들을 마우스 드라이버 소프트웨어에 의해 인터럽트 된다.

그림 3는 애플 표준 마우스 메카니즘을 보여준다. 이 그림에서 구형파 신호들은 마우스가 아래로 이동될 때 발생된다.

마우스의 이동정보는 마우스에 있는 ADB 트랜시버내의 레지스터 0에 저장된다. 애플 표준 마우스내의 레지스터 0의 내용은 표 2 에 보인다.

마우스내의 ADB 트랜시버는 컴퓨터내의 ADB 트랜시버에 데이터를 전송하고, 컴퓨터내의 시프트 레지스터에 전송한다. 마우스 드라이버는 VIA 시프트 레지스터에서 그 정보를 읽어들여 그 데이터를 해석하고 그것을 스크린상에서 적당한 커서 이동으로 바꾼다. 제어패널상에 마우스 스케일링 옵션을 사용하면 유저는 주어진 마우스 이동에 일치하는 스크린 표인터에

〔그림 3〕 매킨토시 SE 애플 표준 마우스 메카니즘



〔표 2〕 애플 표준 마우스의 레지스터 0의 내용

비 트	의 미
15	Button status; 0 = down
14-8	Y move counts
7	Not used (always 1)
6-0	X move counts

이동의 양을 마우스가 얼마나 빨리 이동하느냐에 따라 커서위치를 신속히 바꿀 수가 있다.

애플 표준 키보드

애플 표준 키보드는 기존의 매킨토시 키보드에 설치되어 있는 모든 키들을 갖고 있으며, 새로운 두가지 키 즉, Eacape키와 Control키가 추가되어 있다.

키보드 배열과 Key-Down 천이코드(Transition Code)들을 **그림 4** 나타내었다. Key-Up 천이코드들은 비트7이 1로 세트된 것을 제외하고는 Key-Down 천이코드들과 똑같다.

이러한 응답코드들은 키보드 드라이버 소프트웨어에 위해 리턴되는 키 코드들과는 전부 똑같지는 않다는데 유의해야 한다. System Folder에 공급되는 표준 'KMAP' 리소스(Resource)는 **표 3**에 보인 것처럼 여러 키 코드들을 재할당(Reassign) 한다.

키보드 정보는 애플 표준 키보드에 있는 ADB 트랜시버내의 레지스터 0에 저장된다. 두가지 키의 상태들이 키보드 레지스터 0에 동시에 저장될 수 있다. 또한 변경 키(Modifier Key)들의 상태들도 키보드내의 레지스터 2에 저장된다.

ADB 매니저들로부터의 명령에 따라 애플 표준 키보드내의 ADB 트랜시버는 레지스터 0에 있는 데이터를 컴퓨터내의 ADB 트랜시버로 전송하며, 이것은 그 데이터를 VIA 시프트로 레지스터 전송한다. 그래서 ADB 매니저는 제어를 키보드 드라이버로 보내고 키보드 드라이버는 모든 키보드 정보를 읽어 들여, 그것을 해석한 다음 응용에 가능한 형태로 만든다.

애플 확장 키보드

애플 확장 키보드는 매킨토시 오퍼레이팅 시스템과는 다른 오퍼레이팅 시스템을 수행하기를 원하는 유저를 위해 주로 설계되어 있다. 이러한 키보드는 또한 응용 프로그램과 기능 키보드를 사용하는 다른 컴퓨터에서 전송된 데이터통신 패키지에 유용하다.

애플 확장 키보드는 기존의 매킨토시 키보드에 장치된 모든 키들(또는 이들과 기능면에서 등가인)을 갖고 있으며 여기에 새로운 여러 키들 즉, 디폴트 정의를 갖지 않는 12개의 기능키(F1~F12), 디폴트 정의를 갖는 9개의 기능키와 2개의 수정키들을 갖고 있다. 새로운 수정키들은 Escape키와 Control키이다. 기능 키들은 애플 표준 키보드에는 존재하지 않는다.

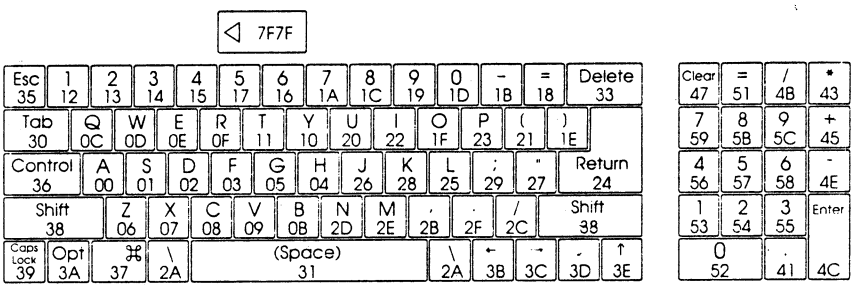
애플 확장 키보드 배열과 Key Down 천이코드(Transition Code)들은 **그림 5**에서 보는 바와 같다. Key-Up 천이코드들은 비트7이 1로 세트된 것을 제외하고는 Key-Down 천이코드들과 똑같다.

그림 5에서 괄호속에 나타낸 키 코드들은 애플 확장 키보드에 레지스터 3내의 디바이스 핸들러 ID의 값이 $0002에서 $0003으로 변할 때 키보드에 레지스터 3 명령을 보냄으로서 바꿀 수 있다.

이러한 응답코드들은 키보드 드라이버 소프트웨어에 의해 리턴된 키코드들과 모두 같지 않다는데 유의해야 한다. System Folder에 의해 공급된 표준 'KMAP' 리소스(Resurce)는 **표 6**에 보인 것처럼 여러 키코드들을 재할당(Reassign) 한다.

키보드 정보는 애플 표준 키보드에 있는 ADB 트랜시버의 레지스터 0에 저장되어 있다. 두 키들의 상태들이 키보드 레지스터 0에 한번에 저장될 수 있다. 또한 수정키와 LED들의 상태들은 키보드내의 레지스터 2에 저장되어

〔그림 4〕 애플 표준 키보드 배열



〔표 3〕 키코드 재할당

키	천이 코드	키보드 드라이버 코드
Control	$36	$3B
Left Arrow	$3B	$7B
Right Arrow	$3C	$7C
Down Arrow	$3D	$7D
Up Arrow	$3E	$7E

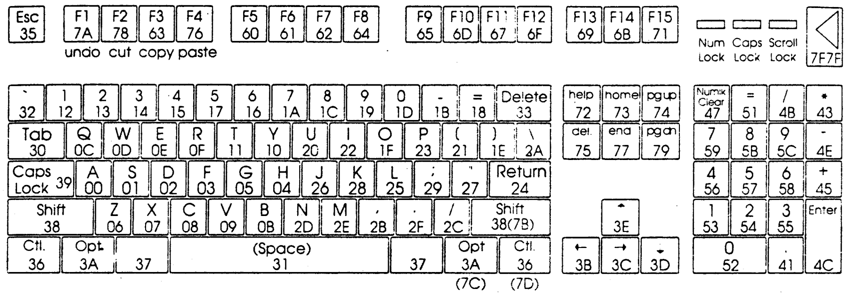
〔표 4〕 애플 표준 키보드 레지스터 0

비 트	의 미
15	Key status for first key; 0 = down
14–8	Key code for first key; a 7-bit ASCII value
7	Key status for second key; 0 = down
6–0	Key code for second key; a 7-bit ASCII value

〔표 5〕 애플 표준 키보드 레지스터 2

비 트	키보드 드라이버 코드
15	None (reserved)
14	Delete
13	Caps Lock
12	Reset
11	Control
10	Shift
9	Option
8	Command (Apple)
7–0	None (reserved)

〔그림 5〕 애플 확장 키보드 배열



〔표 6〕 애플 확장 키코드의 재할당

키	천이 코드	키보드 드라이버 코드
Control	$36	$3B
Left Arrow	$3B	$7B
Right Arrow	$3C	$7C
Down Arrow	$3D	$7D
Up Arrow	$3E	$7E
Right Shift	$7B	$3C
Right Option	$7C	$3D
Right Control	$7D	$3E

있다.

ADB 매니저로부터의 명령에 따라 애플 확장 키보드내의 ADB 트랜시버(Transceiver)는 레지스터 0내에 있는 데이터를 컴퓨터내의 ADB 트랜시버로 전송하며 이것은 그 데이터를 VIA 시프트 레지스터로 전송한다. 그래서 ADB 매니저는 키보드 드라이버에 제어를 보내고 키보드 드라이버는 모든 키보드 정보를 읽어 들여 그것을 해석해서 응용에 가능한 상태로 만든다.

ADB 통신

네트워크에서 각각의 ADB 입력 디바이스는 ADB 디바이스 레지스터라는 4개의 내부 레지스터들을 가질 수 있다. ADB 디바이스들은 디바이스내의 적당한 ADB 디바이스 레지스터로부터 리드하거나 그 레지스터에 라이트함으로서 네트워크상에 액세스하게 되어 있다.

ADB 디바이스의 레지스터들은 2에서 8바이트 범위까지 가질 수 있다.

비록 각각의 ADB 디바이스는 마이크로 프로세서를 포함하고 있고 디바이스 인식과 상태정보를 발생시킬 수 있지만 ADB 디바이스는 명령을 개시할 수 없고 프로세서를 직접 인터럽트할 수 없다. 컴퓨터내의 ADB 트랜시버는 새로운 데이터를 위해 데이터를 보내도록 최종 디바이스를 자동적으로 폴(Poll) 한다. 만일 액티브 디바이스가 보낼 데이터를 갖고 있다면, 그 디바이스는 그 데이터를 컴퓨터내의 ADB 트랜시버 밖으로 시프트하고 ADB 트랜시버는 그 데이터를 VIA에 보낸다. VIA가 ADB 데이터를 받았을 때 VIA 인터럽트 레지스터내의 한 비트는 세트된다. 그래서 ADB 매니저는 액티브 디바이스가 보내온 데이터를 결정할 수 있다. 그 다음 ADB 매니저는 액티브 디바이스에 대한 디바이스 드라이버에 제어를 보내고 그 드라이버는 VIA와 ADB 트랜시버를 통해 디바이스와 통신한다.

액티브 디바이스가 아닌 ADB 디바이스가 보낼 데이터를 갖고 있다면, 그 디바이스는 컴퓨터에 서비스 리퀘스트를 어서트(Assert) 한다. 컴퓨터내의 ADB 트랜시버는 VIA 데이터 레지스터 B내의 비트3을 0으로 세트하여 그의 인터럽트 요구신호를 어서트(Assert) 한다. 매킨토시 오퍼레이팅 시스템은 VIA를 규칙적으로 폴(Poll)하고 이 비트에 0을 발견했을 때 오퍼레이팅 시스템은 ADB 매니저에 제어를 보내고 ADB 매니저는 하나의 요구하는 디바이스를 발견할 때 까지 각 디바이스를 폴(Poll)한다. 그 다음 ADB 매니저는 VIA와 ADB 트랜시버를 통해

디바이스와 통신하는 적당한 디바이스 드라이버에 제어를 보낸다.

ADB상의 통신은 두가지 형태, 즉 Transaction과 Signal 로 이루어져 있다. 트랜젝션(Transaction)은 컴퓨터에서 ADB 디바이스로 흐르는 한 명령과 그 다음에 컴퓨터나 디바이스에 의해 보내진 데이터로 이루어진다. 각 명령은 특정 디바이스에 어드레스된다. 신호들은 디바이스의 상태를 표시하거나 모든 디바이스로 하여금 어떤 기능을 수행하도록 한다. 신호는 일반적으로 특정 디바이스에 어드레스되지 않는다.

다음 섹션에서는 ADB 트랜젝션을 정의하고, 명령들과 신호들을 설명하며, ADB 명령과 신호에 대한 타이밍 규정들을 리스트 한다.

●트랜젝션

ADB 트랜젝션(Transaction)은 컴퓨터와 디베이스사이의 버스 통신이다. 트랜젝션은 컴퓨터에 의해 보내진 명령과 그 다음에 컴퓨터나 디바이스 중의 하나에 의해 보내진 여러 바이트의 데이터 패킷으로 이루어 진다. ADB 명령은 다음으로 이루어진다.

- 한 Attention / Sync 신호
- 한 명령바이트
- 한 스톱비트

한 데이터 패킷은

- 스타트 비트
- 2에서 8 데이터바이트들
- 한 스톱비트

로 이루어져 있다.

ADB 트랜젝션의 각 부분이 완료된 후 컴퓨터는 다음 부분의 통신을 준비하기 위해 다음 트랜젝션 상태로 ADB 트랜시버가 옮겨가도록 라인 ST0와 ST1을 사용한다.

ADB 명령을 수행하기 위해 ADB 매니저는 VIA 시프트 레지스터에 명령 바이트를 보낸 다음 ADB 트랜시버를 상태 0으로 세트한다. 트랜시버는 VIA로부터 명령을 시프트하고 그 명령을 ADB 프로토콜로 바꿔서는 그것을 ADB상에 놓는다. VIA는 전체 바이트가 전달됐을 때 인터럽트를 발생시킨다. 프로세서가 그 인터럽트를 받으면 ADB 매니저는 VIA로부터 ADB 트랜시버로 전달하고 그런 다음 ADB 버스에 데이터를 전달하거나 디바이스로부터 트랜시버로 데이터를 전달한다. 그런 다음 VIA(Take 명령을 위해)에 전달하기 위해 1과 2사이의 트랜젝션 상태를 바꾼다.VIA는 각 바이트가 전달된 후에 인터럽트를 발생시킨다.

데이터의 최종 바이트가 전달된 후에 ADB 매니저는 다른 명령을 보낼려고 원한다면 트랜젝션 상태를 0으로, 이 상태 중에서 ADB 트랜시버가 매 11ms마다 최종 Talk 명령을 자동적으로 반복한다면 3으로 세트한다. 부분적으로 수행된 명령을 포기하기 위해서는 ADB 매니저는 ADB 트랜젝션 상태를 0으로 세트한다.

스타트업(Start up)이나 리세트에 대한 디폴트 트랙젝션 상태는 3이다.

ADB 주변장치

ADB상의 모든 주변장치 디바이스들은 슬레이브(Slave)이며 단지 컴퓨터만이 명령을 보낼 수 있다. ADB 디바이스들은 컴퓨터로 부터 Talk 명령을 받은 후에야 버스에 데이터를 전송한다.

각각의 ADB 디바이스는 다음의 사항들을 필요로 한다.

- 데이터를 저장하기 위한 메모리
- 하나의 디폴트 ADB 어드레스
- 하나의 디폴트 ID
- 버스 충돌에 대한 검출과 대처 능력
- 서비스 리퀘스트 신호를 어서트(Assert) 하는 능력

이에 추가하여 어떤 ADB 디바이스들은 하나 이상의 기능모드를 가지며, 디바이스 드라이버는 새로운 디바이스 핸들러 ID를 디바이스에 할당함으로서 그 모드를 선택할 수 있다.

다음 섹션은 ADB 디바이스의 이러한 형태들을 설명한다.

●디바이스 레지스터

ADB 디바이스는 레지스터 0에서 3으로 데이터를 저장할 수 있는 4개의 배치들을 가질 수 있다. 각각의 Talk나 Listen 명령은 특정 레지스터에 직접 관련된다.

각 레지스터의 기능은 다음과 같다.

●레지스터 0

레지스터 0은 데이터 레지스터이다. ADB 매니저는 어떤 디바이스가 서비스 리퀘스트 신호를 보내는지 결정하기 위해 디바이스들을 폴(Poll)할 때 토크 레지스터 0 명령을 교대로 각 디바이스에 보냄으로서 그것을 수행한다. 그러므로 디바이스는 서비스를 요구할 때 비록 그 디바이스 핸들러에 중요한 데이터가 다른 레지스터내에 있을지라도 레지스터 0에 데이터를 갖는게 효율적이다.

●레지스터 1

레지스터 1은 디바이스 특정 데이터 레지스터이다. 이 레지스터는 어떤 데이터 기능을 위해서 디바이스에 의해 사용될 수 있다.

●레지스터 2

레지스터 2는 디바이스 특정 데이터 레지스터이다. 이 레지스터는 어떤 데이터 기능을 위해서 디바이스에 의해 사용될 수 있다.

●레지스터 3

이 레지스터는 상태와 디바이스 표시 정보를 포함한다. 어떤 디바이스를 위해 컴퓨터는 Listen Register 3 명령을 갖는 이 레지스터를 어드레싱함으로서 디바이스의 기능을 바꿀 수 있다.

●디바이스 어드레스

각각의 ADB 디바이스는 디폴트 4-비트 버스 어드레스를 갖는다. 어떤 디바이스 형태들은 표 7에 보인 것처럼 특정 디폴트 어드레스들을 갖는다. 예를 들어 마우스와 같은 모든 상태 위

〔표 7〕 디바이스 어드레스

어드레스	디바이스 형태	예
$00–$01	Reserved	–
$02	Encoded devices	Keyboard
$03	Relative devices	Mouse
$04	Absolute devices	Graphics tablet
$05–$07	Reserved	–
$08–$0F	Any other	

치 디바이스들은 어드레스 3에서 파워 업(Power Up) 된다. 대부분의 디바이스들은 이동가능한 어드레스들을 갖고 있는데, 즉 컴퓨터는 새로운 어드레스를 디바이스에 할당할 수 있다. 이것은 두개의 디바이스(가령 마우스 디바이스와 상대위치 그래픽 타블렛과 같은)들이 똑같은 디폴트 어드레스를 가질 때 하나는 새로운 어드레스로 이동되어야 하기 때문에 필요하다. 현재, 8개의 어드레스들은 다른 형태의 디바이스들에 가능하도록 8개의 디폴트어드레스들을 남겨놓기 위해 미리 지정되어야 유보된다.

●Device handler ID

ADB 매니저는 8비트 디바이스 핸들러 ID 디폴트 ADB 어드레스와 함께 –를 어떤 디바이스 드라이버가 특수 디바이스를 호출하는 지를 결정하기 위해 사용한다. ADB 디바이스 드라이버는 특수 ADB 디바이스나 그와 같은 계층의 디바이스들과 통신하는 시스템 소프트웨어 루틴인데, ADB 디바이스 드라이버는 시스템 화일내의 ADBS 리소스(Resource)에 배치되어 있다. 시스템 스타트업(Startup)에서 Start Manager는 ADBS 리소스에 대한 시스템 화일을 찾아서 그것을 메모리에 로드하고서는 그것을 수행한다. Start Manager는 또한 각각의 ADB 디바이스의 레지스터 3을 리드하여 ADB 디바이스 테이블에 디바이스의 디폴트 어드레스와 디바이스 핸들러 ID를 위치시킨다.

각각의 ADBS 리소스는 디바이스 형태 (디바이스 핸들러 ID에 일치하는)와 ADB 어드레스(디바이스의 디폴트 어드레스에 일치하는)를 포함한다. 같은 디폴트 어드레스를 갖는 ADB에 하나 이상의 디바이스가 있는 경우에 Start Manager는 더 이상의 분쟁이 없을 때까지 새로운 어드레스들로 디바이스들을 재할당(Reassign) 한다.

ADB 디바이스 테이블은 각각의 디바이스에 대한 원래의 어드레스와 현재의 어드레스를 모두 포함한다. ADB 매니저는 각 디바이스를 특수 디바이스 드라이버들을 갖는 버스에 관련시키기 위해 디폴트 ADB 어드레스와 디바이스 핸들러 ID를 사용한다.

2개 이상의 디바이스들이 똑같은 디바이스 드라이버를 사용할 수 있다.

디바이스는 하나 이상의 기능모드를 가질 수 있으며, 만일 그렇다면, 디바이스 드라이버는 디바이스 핸들러 ID를 바꾸기 위해 Listen Register 3 명령을 사용할 수 있다. 만일 그 디바이스가 새로운 디바이스 핸들러 ID를 인정한다면, 거기에 따라 기능 모드들을 바꾼다.

유보된 디바이스 핸들러 ID를 일단 받으면 디바이스는 즉시 특정 기능을 수행한다. 디바이스는 레지스터에 유보된 디바이스 핸들러 ID를 저장하지는 않으며, 단지 디바이스정의 핸들러는 코드들만이 저장된다. 모든 인정되지 않은 디바이스 핸들러 ID는 무시된다.

●충돌 검출

각 디바이스의 비어있는 버스를 찾아낼 때까지, 즉 그 버스가 하이상태로 남아있을 때까지 데이터 전송하는 것을 기다린다. 만일 디바이스가 버스를 하이(비트셀을 완성하기 위해)로 가져갈려고 시도하고 다른 디바이스는 그 라인을 로우로 한다거나 다른 디바이스는 그 디바이스가 그의 스타트 비트(Start bit)를 어서트(Assert)하기전에 데이터를 보내는 것을 시작한다면 그 디바이스는 충돌상실을 잃었다고 말할 수 있다. 상실 디바이스는 즉시 전송하는 것을 멈추고 보내고 있었던 데이터를 보호한다.

모든 ADB 디바이스들은 스타트업 동안 유일한 어드레스로 할당되고 한 디바이스는 그 디바이스에 특별히 할당된 Take 명령에 응답하여 데이터를 보내기 때문에 충돌은 보통 발생하지 않는다.

스타트업에서 스타트 매니저는 각각의 ADB 어드레스로 Talk Register 3 명령을 보낸다. 만일 한 어드레스에서 하나 이상의 디바이스가 존재하면, 충돌을 잃은 각 디바이스는 그의 내부 충돌플래그를 세트하고 그의 이동가능한 어드레스 기능을 불가능하게 한다. 그래서 스타트매니저는 $FE의 핸들러 ID와 새로운 디바이스 어드레스를 갖는 그 어드레스에 Listen Register 3 명령을 보낸다. 충돌을 발견하지 않은 디바이스는 새로운 어드레스로 이동된다. 이 과정은 그 어드레스에서 Talk Register 3 명령에 대한 응답이 타임아웃될 때까지, 즉 그 어드레스에서 더 이상의 디바이스들이 없을 때까지 계속된다. 그래서 그 디바이스들 중의 하나는 디폴트 어드레스로 다시 이동되고 스타트 매니저는 다음 어드레스로 계속 진행한다.

ADB 디바이스는 개개의 디바이스들을 인식하거나 배치하기 위해 멀티유저 응용에 사용될 수 있는 키나 버튼을 가질 수 있다. 디바이스 드라이버는 $FD의 디바이스 핸들러 ID를 갖는 Lister Register 3 명령을 불러내어 새로운 어드레스로 디바이스를 재배치할 수 있다. PC

시스템 자원(1)

시스템 환경

프로그램이 로드되었을때 첫번째 일은 프로그램이 어디에 있는지를 찾아내야 한다. 즉, 어떤 종류의 IB-M-PC 가 그 프로그램을 수행하고 있는지? DOS 버전은 얼마인지? 필요한 주변기기가 모두 준비되어 있는지? 이러한 정보들을 알아내는 데는 세가지 방법이 있다.

가장 단순한 방법은 프로그램 사용자에게 이러한 정보를 간단히 암시해 주는 것이다. 그다음 좀더 나은 방법은 시스템 보드상의 딥스위치 세팅으로 부터 가능한한 많은 정보를 얻는 것이다. 그러나 이러한 세팅은 항상 충분한 정보를 우리에게 제공해주지는 않는다. 더 나은 세번째 방법은 의심이 가는 하드웨어에 직접 액세스시키거나, BIOS의 데이터 영역에서 정보를 알아내는 방법이다.

프로그램이 하드웨어를 액세스하는 데는 두가지 방법밖에 없다. 그 하나는 하드웨어에 연결되는 단자 어드레스(65535개의 가능한 단자 어드레스 중에 적은 부분만이 사용됨) 중의 어떤 어드레스로 부터 리드하고 그 어드레스에 라이트하는 방법이다. 다른 또 하나는 랜덤 액세스 메모리 어드레스공간에 존재하는 어드레스중의 어떤 어드레스로 부터 리드하고, 그 어드레스에 라이트하는 방법이다.

그림1은 운영 체제(Operating System) 와 프로그램이 메모리에 어떻게 분배되는 가를 보여준다.

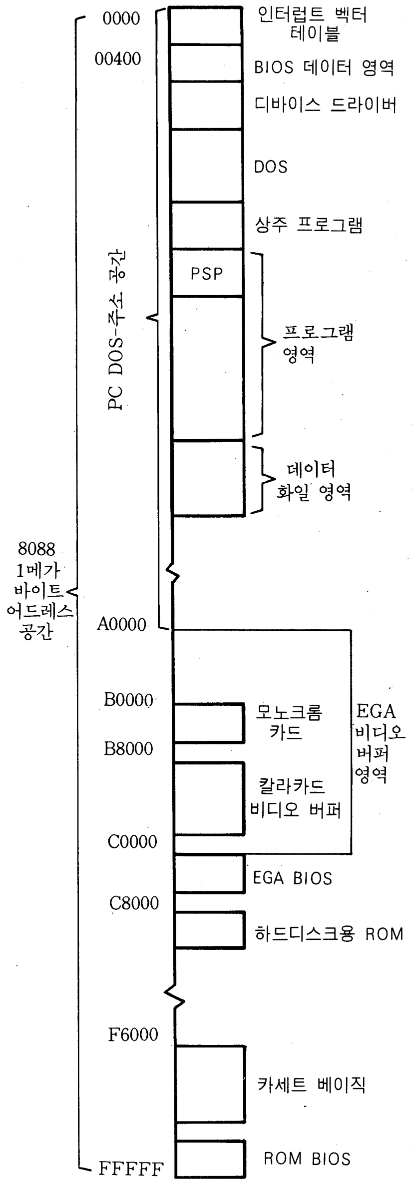
1. 8255주변 인터페이스의 액세스방법

인텔 8255주변 인터페이스 칩은 어떤 주변장치가 있는지에 대한 정보를 알려주는 데 가장 적합하다. 이 칩은 다용도로 사용되고 있는데, 주로 시스템 보드상에 딥스위치를 고정시키는데 도움을 주고, 키보드로 부터 컴퓨터의 입력을 받아들인다. 또한, 8253타이머 칩을 포함하는 일련의 주변장치를 제어한다.

IBM-PC중에 AT는 8255를 사용하지 않으나, 대신에 그 구성정보(Configuration Information)를 백업 전지로 동작하는 특수 칩에 리얼타임클럭에 따라 저장한다. 그러나 AT는 키보드 조작을 위해서나 타이머 칩을 제어하는 데는 똑같은 8255 단자 어드레스를 사용하고 있다. 8255는 포트 A~C로 언급되는 3개의 1바이트 레지스터를 갖는다. 이 레지스터들은 단자 어드레스 60H-62H에 각각 할당된다.

3단자 모두 리드하는데 사용되지만, 포트는 라이트도 겸할 수 있다. PC에서 포트B의 비트7을 1로 세팅하면 포트A는 정지하라는 정보로 변한다. 마찬가지로 PC에서 비트 2로 세팅하면 포트C의 4개의 하위비트 내용이 결정되는데, XT에서는 비트 3으로 세팅하면 이와 똑같은 내용을 행한다. 레지스터의 내용은 다

〔그림 1〕 1MB 어드레스 공간



음 표 1과 같다.

여기서 레지스터 비트중에 0은 딥스위치를"OFF"로 세팅하는데 사용된다.

AT는 그 구성세팅(Configuration Settings)을 모토롤라 MC146818칩으로 리얼타임 클럭에 따라 유지한다. AT에는 타이머 칩을 제어하거나 키보드로 부터 데이터를 받아들이기 위해 단자 어드레스가 사용되지만, 가령 8225와 같은 칩은 없다. 칩은 00－3FH로 번호가 붙여진 64개의 레지스터를 갖는다.

한 레지스터를 읽어 들이기 위해서는 먼저 그 레지스터의 번호를 단자 어드레스 70H에 보낸 다음 71H로 부터 읽어 들인다. 지금까지 설명한 여러가지 구성에 따른 세팅은 다음의 **표2**와 같다.

• 베이직의 경우

다음에 소개하는 베이직 프로그램을 이용하면 PC에 장치된 디스크드라이브의 수를 간단히 알아낼수 있다. 여기서는 포트A는 2개의 상위비트를 읽어 들이기 전에 포트B의 비트 7을 1로 세트시켜 놓으며 기본적으로 프로시딩(Proceeding) 또는 키보드를 잠그기전이나 컴퓨터를 끄기전에 비트를 0로 바꿔 놓아야 한다.

베이직에서는 비트 형식을 분석하는데 어려움을 겪게되는 숫자의 2진 표현이 허용되지 않는다. 간단한 서부루틴은 255(단자가 제공할 수 있는 가장 큰수)까지의 정수를 8캐릭터 2진 스트링으로 나누어 변화시킨다. 따라서 MID$과 같은 스트링 기능(String Function)은 분석하는데 적절한 비트를 추출한다. (리스트 1 참조).

• 어셈블리어의 경우

어셈블리어 프로그램은 베이직에서의 예와 같이 똑같은 방법으로 디스크드라이브의 수를 알아낼 수 있는데 좀더 쉽게 할 수 있다. 이때 사용후에는 반드시 포트B의 원래의

〔표 1〕 레지스터의 내용

포트 A(60H)		
포트 B의 비트 7이 0일 때 : 비트 0~7		키보드에서 8비트를 스캔
포트 B(61H)		
비트	0	타이머 칩 채널 2의 8253의 게이트를 콘트롤
	1	스피커에 출력
	3	포트 C의 내용을 셀렉트
	4	0은 램을 인에이블
	5	0은 확장슬롯의 에러 신호를 인에이블
	6	1은 키보드 클럭 신호를 인에이블
	7	키보드 인식
포트 C(62H)		
XT상에서 포트 B의 비트 3이 1일 때 : 비트	1	1은 수치 연산코프로세서를 인스톨
	2~3	시스템보드상의 램 뱅크 시스템보드
	4	사용되지 않음
	5	8253 채널 2의 슬롯 출력
	6	1은 확장슬롯 에러 체크
	7	1은 패리티 체크
XT상에서 포트 B의 비트 3이 0일 때 :	0~1	화면 형태
	2~3	디스크드라이브 수(00＝1, 등)
	4~7	포트 B의 비트 2가 1인 경우와 같다.

〔리스트 1〕

```
100 A=INP (&H61)                         '포트 B안의 값을 얻는다.
110 A=A OR 128                           '비트 7이 사용된다.
120 OUT &H61 ,A                          '포트 B에서 바이트를 다시 준다.
130 B=INP (&H60)                         '포트 A안의 값을 얻는다.
140 A=A AND 127                          '비트가 바뀐다.
150 OUT &H61,A                           '포트 B로 원래의 값을 재저장
160 GOSUB 1000                           '이진 문자열로 변환한다.
170 NUMDISK$=RIGHT$(B$,1)                '비트 0를 얻는다.
180 IF D$="1" THEN NUMDISK=0:GOTO 230    '디스크 시스템이 없다.
190 C$=LEFT$(B$,2)                       '문자열의 2개의 상단 비트를 취한다.
200 TALLEY=0                             '디스크 번호의 표시를 유지한다.
210 IF RIGHT$(C$,1) = "1" THEN TALLEY=2 '하위 비트 구성
220 IF LEFT$(C$,1) = "1" THEN TALLEY = TALLEY+1 '하위 비트를 더한다.
230 TALLEY=TALLEY+1                      '0이 아니라 1에서부터 계산
                                         '이제 드라이브 번호를 갖게 된다.

1000 '''바이트를 바이너리 문자열로 바꾸는 서브루틴
1010 B$=""                               'B$는 문자열
1020 FOR N=7 TO 0 STEP-1                 '2의 멱승보다 더 작은가 테스트
1030 Z=B-2^N                             '바이트의 값에서 빼기
1040 IF Z>=0 THEN B=Z:B$=B$+"1" ELSE B$=B$+"0" '문자열 모집
1050 NEXT                                '각 비트를 반복
1060 RETURN                              '처음부터 다시
```

값으로 되돌려 놓아야 한다 (리스트 2 참조).

2. IBM-PC의 기종을 아는 방법

여러 종류의 IBM-PC사이에는 호환성 문제가 대두된다. 프로그램이 어떤 IBM 기종에서 수행되어 그 기종의 기능을 충분히 이용하고자 할 때는 프로그램이 로드되는 기종을 알아내야만 한다. 이러한 정보는 메모리 공간의 최종 바이트중의 2번째인 BIOS ROM의 어드레스 FFFFEH 에서 알아낼 수 있으며 다음의 **표3**과 같은 코드 번호를 사용한다.

• 베이직의 경우

베이직에서는 값을 읽어 들이기 위해 간단히 PEEK를 사용한다 (리스트 3 참조).

• 어셈블리어의 경우

어셈블리어 언어에서는 다음과 같이 한다(리스트 4 참조).

3. 비디오 어댑터의 번호와 종류를 알아내는 방법

시스템을 사용할때 현재 시스템의 상태가 모노크롬 보드에서 수행되고 있는지, 또는 칼라보드나 EGA에서 수행되고 있는지, 보조 어댑터가 존재하는지를 알아야 할 경우가 있다. 이럴 경우는 BIOS 데이터영역의 상태바이트를 체크하면 된다.

BIOS 데이터 영역의 0040:0010에 있는 장치 상태바이트(Equipment Status Byte)는 딥스위치 1에 세팅을 하여 어떤 카드로 수행할 것인가를 결정하는 역할을 한다. 이상적으로 비트 5-4는 만일 단색카드일 때는 11로, 칼라 카드중 80×25에서는 10으로, 칼라카드중 40×25에서는 01로, EGA일 때는 00로 세팅한다. 그러나, EGA가 존재한다면, EGA의 스위치가 세트된 상태에 따라 00보다는 다른 값으로 비트를 세트시킬 수도 있다.

사용자가 EGA가 존재하는지 안하는지를 알아내기 위해 다른 수단을 사용해야 하지만, 그렇지 않다면 BIOS 데이터는 사용 어댑터가 단색카드인지, 칼라 카드인지를 표시하게 된다. EGA의 존재여부에 대한 검사를 하기 위해서는 0040:0087에 있는 바이트를 테스트한다. 만일 그 바이트가 0이면 EGA는 존재하지 않고, 0이 아닌 상태에서 비트 3이 0이면 EGA는 사용 어댑터에 해당

〔표 2〕 구성에 따른 세팅

레지스터 번호	사 용
10H	플로피디스크 드라이브 형태
12	하드디스크 드라이브 형태
14	주변기기
15	시스템 보드 메모리(로우 바이트)
16	시스템 보드 메모리(하이 바이트)
17	토탈 확장 메모리(로우 바이트)
18	토탈 확장 메모리(하이 바이트)
30	1MB이상 확장 메모리(로우 바이트)
31	1MB이상 확장 메모리(하이 바이트)

〔리스트 2〕

```
IN   AL,61H              ;포트 B안의 값을 얻는다.
OR   AL,10000000B        ;비트 7에 1을 준다.
OUT  61H,AL              ;바이트를 재배치
IN   AL,60H              ;포트 A안의 값을 얻는다.
MOV  CL,6                ;오른쪽의 AL을 쉬프트하기 위해 셋업시킴
SHR  AL,CL               ;상단 2비트를 쉬프트하고 6을 오른쪽에 위치시킴
INC  AL                  ;0이 아니라 1에서부터 계산
MOV  NUM DRIVES,AL       ;이제 드라이브 번호를 갖게된다.
IN   AL,61H              ;포트 B를 재저장하기 위해 준비
AND  AL,01111111B        ;비트 7이 바뀐다.
OUT  61H,AL              ;바이트를 재배치
```

〔리스트 3〕

```
100 DEF SEG=&HF000       ;메모리의 상단을 64K로 지시한다.
110 X=PEEK(&HFFFE)       ;마지막 바이트로부터 두번째로 얻는다.
120 IF X=&HFE THEN...    ;그러면 XT이다.
```

〔리스트 4〕

```
;---컴퓨터 타입 찾기:
        MOV  AX,0F000H           ;롬으로 ES를 지시
        MOV  ES,AX               ;
        MOV  AL,ES:[0FFFEH]      ;바이트를 얻는다.
        CMP  AL,0FEH             ;XT인가?
        JE   INITIALIZE_ JR      ;그렇다면 초기의 코드로 goto
```

되고 비트 3이 1이면 보조 어댑터에 해당된다.

EGA상태에서는 6845 칩에다 커서어드레스 레지스터에 해당하는 값을 써 넣는 방법에 의해 단색 카드인지 칼라 카드인지를 찾아 낸 다음 그 값을 읽어 들여 그 카드가 적합한지를 살펴본다. 단색 카드에서는 커서 레지스터에 색인을 달기위해 0FH를 단자 3B4H에 보낸다음 단자 3B5H로 부터 커서 어드레스를 리드와 라이트한다. 칼라카드일때 원색카드와 대응되는 단자는 3D4H와 3D5H 이다.

카드가 존재하지 않을 때 단자들은 0FFH에 리턴되지만, 이러한 번호는 레지스터에 의해 유지되기 때문에 그 값으로 테스트를 하는 것은 충분하지 않다.

EGA에 대해서 사용자가 해결해야 할 다른 두가지의 문제가 있다. 즉, 얼마의 메모리가 존재하는지? EGA에 연결되는 모니터의 기종은 무엇인지? 디스플레이의 기종을 알기 위해 0040:0087에 있는 비트 1을 테스트하는데, 그것이 1이면 단색 디스플레이가, 0이면 칼라 디스플레이가 연결된 것이다.

만일 사용자의 프로그램이 350라인 칼라 보드를 사용한다면, 칼라 디스플레이가 IRGB인지 R'G'B' RGB 인지를 나타낼 필요가 있는데, 여기서 후자는 IBM Enhanced Color Display와 일치한다. 이것은 EGA 그 자체에 있는 4개의 딥스위치의 세팅에 의해 조작되어 진다. 이러한 세팅은 INT 10H의 기능 12H가 호출될 때 CL로 리턴된다. Enhanced Color Display에 있어서 하위 4비트의 형식은 0110가 된다. 이와 같은 기능은 EGA에 있어서 메모리의 양을 나타낸다. 리턴에서 BL은 비디오램이 64K일 때 0, 128K일 때 1, 192K일 때 2, 256K일 때 3을 포함한다.

• 베이직의 경우

이러한 코드요소는 현재의 모니터 기종과 모드를 검사하고, 이 모니터에 어떤 종류의 영상 어댑터가 내장되어 있는지를 알아낸다 (리스트 5 참조).

이 예는 EGA 또는 칼라카드가 동작할 때 단색 카드에 대해 검사를 한다. 동일 코드로 만일 사용자가 단자 어드레스 &H3D4와 &H3D5를 사용하고 있다면 칼라카드를 찾아낼 수 있다(리스트 6 참조).

• 어셈블리어의 경우

이 예는 위에서 나타낸 베이직 예와 동일한 것이다 (리스트 7 참조).

단색카드를 찾았다고 가정하면, 칼라카드(Non-Active)가 내장되어 있는지를 알 수 있다(리스트 8 참조).

4. 디스크드라이브의 번호와 기종을 알아내는 방법

AT를 제외한 모든 컴퓨터에 있는 8255 주변 인터페이스 칩의 레지스터는 플로피디스크 드라이브가 얼마나 많이 컴퓨터에 설치되어 있는가에 대한 정보를 갖고 있다. 디스크의 기종을 표시하는 정보는 디스크 화일 배치표(File Allocation Table; FAT)에 있는데, 이 표는 디스크 공간 및 용도를 나타낸다. FAT에서 첫째 바이트는 다음 표 4와 같은 코드를 갖는다.

화일 배치표는 그 자체가 화일이 아니며 특수 디스크섹터를 직접 읽어들이는 BIOS 또는 DOS기능을 사용하여 리드는 가능하다.

BIOS 데이터는 딥스위치가 단지 플로피에 세트되어 있기 때문에 얼마나 많은 디스크가 놓여 있는지를 나타내지는 않는다. 그러나, 드라이브 찾기 위해서는 여기에 주어지는 운영체제 기능을 사용하면 된다. 앞표에 나온 코드대신에 드라이브가 존재하지 않으면 0CDH로 리턴시킨

〔리스트 5〕

```
100 '''컨트롤에 무슨 어댑터가 있는지 찾기:
110 DEF SEG=&H40           'BIOS 데이터 영역의 시작을 지시
120 X=PEEK(&H87)           'EGA 체크
130 IF X=0 THEN 200        'EGA가 없으면 앞으로 점프
140 IF X AND 8=0 THEN...   'EGA는 컨트롤 안에  있다.


200 X=PEEK (&H10)          '상태바이트를 얻는다.
210 Y=X AND 48             '비트4와 5의 값을 결합
220 IF Y=48 THEN...        '모노크롬은 00110000
230 IF Y=32 THEN...        '80*25은 그래픽 00100000
240 IF Y=16 THEN...        '40*25은 그래픽 00010000
```

〔리스트 6〕

```
100 '''모노크롬 카드가 있을 경우 찾아내기:
110 OUT &H3B4 ,&HF                 '커서 레지스터의 주소
120 X=INP (&H3B5)                  '리드하고 값을 저장
130 OUT &H3B5,100                  '레지스터에 임의의 값을 전송
140 IF INP(&H3B5)<> 100 THEN...    '만약 같이 리턴될 경우는 카드가 있다.
150 OUT &H3B5,X                    '카드가 있으면 값을 재저장
```

〔리스트 7〕

```
;---컨트롤에 무슨 어댑터가 있는지 찾기:
                MOV  AX,40H              ;BIOS 데이터 영역에 ES를 지시한다.
                MOV  ES,AX               ;
                MOV  AL,ES:[87H]         ;EGA가 있나 알아본다.
                CMP  AL,0                ;
                JE   NO_EGA              ;만약 0040:0087이 0이면 EGA가 없다.
                TEST AL,00001000B        ;EGA가 있으면 테스트 비트는 3
                JNZ  EGA_NOT_ACTIVE      ;만약 비트 3이 1이면 EGA는 활성화
                  .                      ;되지못한다.
                  .
EGA_NOT_ACTIVE: MOV  AL,ES:[10H]         ;비디오 상태바이트를 얻는다.
                AND  AL,00110000B        ;비트4와 5를 분리한다.
                CMP  AL,48               ;모노크롬 카드인가?
                JE   MONOCHROME          ;그렇다면 점프
                                         ;아니면 칼라카드로 가정한다.
```

〔리스트 8〕

```
;---불활성 칼라카드가 인스톨 되었는가?:
          MOV  DX,3D4H          ;6845 주소 레지스터를 지시한다.
          MOV  AL,0FH           ;커서 레지스터를 요구한다.
          OUT  DX,AL            ;레지스터를 색인한다.
          INC  DX               ;데이터 레지스터를 지시한다.
          IN   AL,DX            ;현재 리드 하게한다.
          XCHG AH,AL            ;값을 저장
          MOV  AL,100           ;테스트 값으로 100을 사용
          OUT  DX,AL            ;전송한다.
          IN   AL,DX            ;다시 리드
          CMP  AL,100           ;비교
          JNE  NO_CARD          ;카드 없으면 점프
          XCHG AH,AL            ;아니면 칼라 카드이다.
          OUT  DX,AL            ;...그러면 초기 리드를 재저장
```

〔표 3〕 컴퓨터 종류에 따른 코드번호

컴퓨터	코드
PC	FF
XT	FE
PCjr	FD
AT	FC

〔표 4〕 FAT

코드	디스크 형태
FF	양면, 8-섹터
FE	단면, 8-섹터
FD	양면, 9-섹터
FC	단면, 9-섹터
F9	양면, 15-섹터(고밀도)
F8	하드디스크

다. 간단히 이러한 값이 나올때까지 더 높은 드라이브 번호로 계속 테스트를 해나간다.

AT는 그 구성 정보가 어떤 종류의 디스크드라이브를 사용하는지를 표시한다는 점에서 단순하다. 이 정보는 레지스터 번호를 70H로 보낸 다음 단자 어드레스 71H로 부터 얻어낼 수 있다. 디스크에 대한 레지스터 번호는 10H이다. 제1 디스크에 대한 정보는 비트 7-4에, 제2 디스크는 비트 3-0에 들어 있다. 양쪽 모두 비트 형식은 만일 드라이브가 없다면 0000로, 더블사이드(48TPI) 드라이브이면 0001로, 고밀도(96TPI: Track Per Inch) 드라이브이면 0010로 한다. 레지스터에서 하드디스크 정보는 12H이다. 다시 말해 비트 7-4와 3-0은 제 1및 제 2드라이브를 나타내며, 0000는 드라이브가 없는 것을 표시한다. 드라이브의 크기 및 구조를 나타내기 위해 15개의 다른 값으로 분류되지만, 이러한 코드는 복잡하므로 이러한 정보를 필요로 하는 사용자는 AT 기술 참고서적을 참고하기 바란다.

• 어셈블리어의 경우

INT 21H의 기능 1CH는 특수 드라이브에 대한 정보를 나타낸다. 따라서 드라이브 번호를 DL에 놓아야 한다. 여기서,0=디폴트, 1=A 등. DX는 FAT에서 클러스터(Cluster)의 수를, AL은 클러스터 당 섹터의 수를, CX는 섹터에서 바이트의 수를 나타낸다. DS:BX는 앞의 표와 같은 FAT에서 디스크 식별코드를 포함하는 바이트를 나타낸다. 다음의 예는 드라이브 A의 디스크 기종을 알아낸다(리스트 9 참조).

AT BIOS는 일반 드라이브 파라미터를 알려주는 기능을 갖는다. 이것은 INT 13H의 기능 8이다. 또한, DL에 드라이브 번호를, DH에 드라이브중의 가장 큰 번호를, CL에 섹터의 최대 번호를, CH에 트랙을, AH에 디스크 에러상태 코드를 옮긴다.

또 다른 AT BIOS기능은 디스크 형태를 표시한다. 이것은 INT 13H의 기능 15H이다. 여기서 DL은 드라이브 번호를 나타낸다. 이 기능은 AH에 코드를 옮기는데, 여기서, 0=디스크 없음, 1=변화 검출 없는 디스크, 2=변화 검출이 있는 디스크, 3=하드 디스크를 나타낸다. 만일 하드

디스크라면, CX:DX는 512바이트 섹터의 번호를 표시한다.

5. 주변기기를 알아내는 방법

시작점에서 BIOS는 어떤 장비가 연결되어 있는지를 조사한다음 그 조사내용을 상태 레지스터에 보낸다. 그 레지스터는 2바이트의 길이로 0040: 0010에서 시작한다. 다음과 같은 비트 형식은 달리 표현하지 않는한 모든 컴퓨터에 응용할 수 있다. (리스트 10 참조).

대부분의 정보는 직접적이다. 그러나, 디스크 드라이브에 대한 정보는 비트0와 비트 6−7로 분류됨을 유의해야 한다. 6−7에서 0값은 하나의 디스크 드라이브가 존재한다는 것을 표시하므로 존재하지 않은 것을 나타내기 위해서는 비트 0을 고려해야 한다.

부착된 직렬 단자의 번호는 BIOS 데이터 영역을 조사함으로써 알아낼 수 있다. BIOS는 베이스 어드레스를 4개의 COM단자 (DOS는 이것을 단지 2개만 사용함)에 올리기 위해서는 4개의 2바이트 영역을 할당한다. 베이스 어드레스는 COM 채널을 액세스하는 단자의 그룹중 가장 낮은 번호의 단자 어드레스이다. 4개의 영역은 0040:0008에서 시작된다. 즉, COM1은 :0008에서, COM2는 :000A에서 시작된다. 그 영역은 직렬단자와 일치하지 않을 때는 0을 포함한다. 만일 :0008에서의 워드가 0이 아니고 2바이트 높은 :000A에서의 워드가 0이면 하나의 직렬단자가 존재한다.

AT는 주변기기의 정보를 그 구성 칩의 레지스터 14H에 유지시킨다. 처음에 단자 어드레스 70H 밖으로 14H를 라이트한 다음 71H에서 레지스터를 리드한다. 비트 형식은 다음과 같다(리스트 11 참조).

〔리스트 9〕

```
;---디스크 형태 찾기:
        MOV  AH,1CH      ;펑션
        MOV  DL,1        ;드라이브 A를 셀렉트
        INT  21H         ;정보를 얻는다.
        MOV  DL,[BX]     ;드라이브 형태를 얻는다.
        CMP  DL,0FDH     ;양면,9섹터인가?
        JE   DBL_9       ;...기타등등
```

〔리스트 10〕

```
비트 0      1이면 디스크드라이브가 있다.
1           1이면 수치연산용 코프로세서
2-3         11이면 램베이스 64K
4-5         활성되 비디오 어댑터
              (11-08x25 모노크롬카드, 10-08x25 칼라카드, 01-40x25 칼라카드)
6-7         디스크드라이브 번호(비트 0가 1일때)
8           사용하지 않음
9-11        직렬 어댑터의 번호
12          1이면 게임 포트를 부착
13          사용하지 않음
14-15       병렬 어댑터의 번호
```

〔리스트 11〕

```
비트 7-6    00: 1 플로피 드라이브, 01: 2 플로피 드라이브
     5-4     01: 칼라카드 40컬럼 화면
             10: 칼라카드 80컬럼 화면
             11: 모노크롬 화면
     3-2     사용되지 않음
       1     1이면 수치연산용 코프로세서가 인스톨
       0     0이면 디스크드라이브가 없음
             1이면 디스크드라이브가 인스톨
```

- **베이직의 경우**

베이직에서는 간단히 BIOS 데이터 영역으로 부터 직접 상태바이트를 읽어 들인다. 즉 상태 레지스터의 낮은 바이트가 기수인가 우수인가(우수=드라이브 없음)를 확인함에 따라 어떤 디스크드라이브가 존재하는 지를 조사할 수 있다 (리스트 12 참조).

- **어셈블리어의 경우**

BIOS INT 11H는 AX에 장치 상태바이트를 옮기며, 입력레지스터는 존재하지 않는다. 다음의 예는 디스크 드라이브의 번호를 조사하기 위한 것이다(리스트 13 참조).

어셈블리 프로그램 수행은 앞에서 보인 BASIC과 같이 행한다. 다음의 예는 연산 코프로세서 칩이 장치되어 있는 지를 조사하면서, AT에 있어 구성정보를 읽어 들인다 (리스트 14 참조).

6. 램의 할당

"메모리는 얼마인가?"라는 것은 세가지를 의미할 수 있다. 즉 어느 정도의 램이 시스템보드 딥 스위치들 상에 기록되어 있는가? 어느정도의 램 칩의 뱅크들이 컴퓨터내에 존

〔리스트 12〕

```
100 DEF SEG=0                   ;메모리의 최하단을 지시한다.
110 X=PEEK(&H410)               ;레지스터의 하위비트를 준다.
120 IF X MOD 2=0 THEN 140       ;만약 2로 나누어 나머지 없으면 드라이브 없다.
130 PRINT"Disk drive(s) present ;아니면 드라이브가 있고, 메시지를 준다.
140 GOTO 160                    ;2번째 메시지로 점프
150 PRINT"Disk drive(s) absent  ;드라이브가 없으면 메시지를 준다.
160 ...                         ;계속


100 DEG SEG=40H                 ;BIOS 데이터 영역을 시작하기 위해 지시한다.
110 PORT=PEEK(0)+256*PEEK(1)    ;오프셋 0에서 워드를 얻는다.
120 IF PORT = 0 THEN ...        ;COM1 어댑터가 없다.
```

〔리스트 13〕

```
;---디스크드라이브의 번호를 주기
      INT  11H             ;상태 바이트를 얻는다.
      TEST AL,0            ;어떤 드라이브가 있는가?
      JZ   NO_DRIVES       ;아무것도 없으면 앞으로 점프
      AND  AL,1100000B     ;비트5와 6을 분리
      MOV  CL,5            ;쉬프트 레지스터를 오른쪽에 준비
      SHR  AL,CL           ;오른쪽 다섯 비트를 쉬프트
      INC  AL              ;+1하고 1에서 계산시작
                           ;이제 드라이브 번호는 AL안에 있다.
```

〔리스트 14〕

```
MOV  AL,14H          ;레지스터 번호
OUT  70H,AL          ;레지스터요구를 전송
IN   AL,71H          ;레지스터 리딩
TEST AL,10B          ;테스트 비트1
JZ   NO_COPROCESSOR  ;세트되지 않았으면 코프로세서가 없다.
```

재하는가? 프로그램을 할당하는 DOS를 위해 어느정도의 메모리가 실제로 점유되거나 유효한지? 컴퓨터는 640K 정도의 용량을 보유하게 되지만, 이 용량의 1/2정도는 어떤 특수 사용을 위해 따로 확보될 수 있도록 딥 스위치들은 320K정도로 세트될 수 있어야 한다. 그리고, 320K의 가용(Available) 메모리중에서, 메모리의 최상위나 최하위에 로드되거나 상주하고 있는 것이 어떤 다른 소프트웨어인지를 실행 프로그램이 어떻게 알아 낼 수 있을까?

각각의 질문에 대한 대답은 다르게 된다. IBM-PC에서 딥 스위치 세팅들은 8255 주변장치 인터페이스 칩의 포트 B로부터 간단히 리드된다. BIOS는 이용가능한 메모리를 K의 수로 표시하도록 0040:0013에서 2바이트의 가변길이를 갖는다.

AT는 메모리에 대한 매우 우수한 정보를 제공한다. 구조정보를 갖는 칩에서 레지스터 15H(High)와 16H(Low)는 시스템 보드내에 얼마정도의 메모리가 내장되어 있는지(단지3가지의 유효크기 즉, 256K 에서는 0100H, 512K에서는 0200H, 그리고 512K에다 128K메모리 확장옵션을 추가했을 경우는 0280H가 있다.)를 나타낸다. AT내의 I/O채널 메모리는 레지스터 17H와 18H(512K 증가로 주어지는)에 의해 표시된다. 1메가 바이트 범위이상에 위치하는 메모리는 30H와 31H(512 K 증가로 15메가 바이트까지)로부터 가능하다. 만일 AT의 128K 메모리 확장 옵션이 내장되어 있다면, 비트7은 레지스터33에서 1로 세트된다. 모든 경우에 있어 맨처음 레지스터 번호가 단자 어드레스 70H로 보내지고, 그다음 71H로부터 레지스터 값을 리드한다.

메모리 공간내의 일정간격에 램이 존재하는지의 여부를 직접 테스트하는 루틴을 쉽게 라이트할 수 있다. 램이 최소 16K 단위로 내장되므로 단지 전체 16K가 존재한다는 것을 나타낼 수 있는 각 16K 세그먼트내에 한 메모리 배치를 조사하는 것이 필요하다. 메모리 공간내에 한 어드레스가 비어있을 때, 이 어드레스는 233으로 리드되어 진다. 233이 아닌 임의의 번호가 라이트되면 똑같은 어드레스가 그 임의의 번호가 존재하는 지를 판단하기 위해 즉시 리드된다. 233 대신에 임의의 번호가 나타나면 특수 메모리 뱅크는 존재하지 않는다. 메모리 뱅크에 대한 체크를 할 때는 메모리의 내용들내에 변화를 주어서는 안된다. 맨처음(이미 가정된) 메모리 위치의 내용들을 세이브한 다음, 그것을 체크하고, 그다음 원래의 내용들을 복원하는 절차를 밟아야 한다.

프로그램내에 이미 루틴이 위치하게 되면 코드 세그먼트가 사용한 것과 똑같은 옵셋(offset)을 블록내의 옵셋으로서 사용한다. 예를 들어, 코드 세그먼트 레지스터가 12E2를 갖

고 있다면, 세그먼트는 메모리의 두 번째 64K 블록내의 옵셋 3E2에서 시작한다. 실행 루틴은 이 어드레스보다 약간 벗어나서 위치하게 되므로, 각 블록내의 3E2에서 바이트를 체크하는 것이 안전하다. 비가용 인터럽트(Disable Interrupt)들은 체크가 이루어지는 동안 발생하는 하드웨어인터럽트를 위해 코드를 변하게 한다. DOS가 트릭을 요구하는데 얼마정도의 메모리가 허용될 수 있는지를 알아내야 한다. 프로그램이 최초로 제어를 받아 들였는 때 DOS는 프로그램에 DOS의 천이 부분(Transient Part)(이것은 자동적으로 중첩되어 있다면 자동적으로 리로드(Reload)된다.)을 갖는 상위 메모리의 부분을 포함하여 모든 가능한 메모리를 제공한다. 현재 실행되고 있는 프로그램내로부터 다른 프로그램을 실행시키거나 멀티유저 시스템에 적합한 프로그램을 구성하기 위해서는 할당된 메모리를 필요한 크기로 줄여야만 한다. 이러한 변화가 INT21H 의 기능 4AH를 사용하여 어떻게 이루어지는 지를 설명하였다.

이와 똑같은 기능은 할당된 메모리를 확장시키는데에도 사용될 수도 있다. 프로그램이 일단 로드되면 모든 메모리가 할당되므로 그와 같은 확장은 스타트업(Start up)에서는 불가능하다. 만일 그와 같은 확장을 시도한다면, 캐리 플래그(Carry Flag)는 에러 조건(Error Condition)을 표시하기 위해 세트되고, AX내에 에러코드 8이 나타나며, 그리고 16바이트 패러그래프의 가능한 최대수는 BX로 리턴되는데 바로 이것이 필요한 정보이다. BX내에 불가능한 정도의 대형 블록에 대한 요구(즉, F000H 패러그래프)를 하면, 인터럽트가 수행된다.

• 베이직의 경우

베이직은 단지 64K만을 사용한다. (비록 PEEK와 POKE는 64K이외의 메모리를 액세스 할 수 있지만) 가능한 64K의 양은 FRE기능에 의해 리턴된다. 그 기능은 숫자나 스트링일 수도 있는 더미 아규먼트(Dummy Argument)를 항상 취한다. BYTES=FRE(XS)는 비어있는 바이트의 수를 BYTES에 제공한다. BYTES=FRE(X)도 똑같은 일을 수행한다. 그러나, 스트링 아규먼트(String Argument)는 바이트 카운트가 리턴되기 전에 데이터 영역을 깨끗이 비우도록 한다. 작업 공간의 크기가 CLEAR를 사용하여 세트된다면, FRE에 의해 표시된 그 양은 인터프리터 작업영역의 요구에 따라 2.5에서 4Kbyte이하가 될 것이다.

IBM 베이직컴파일러는 코드 및 합성된 데이터상에 64K 제한을 두지 않는다. 그러나 컴파일러 그 자체는 컴파일되는 동안 얼마정도의 메모리를 사용할 수 있느냐에 따라 제한된다. 만일 메모리 공간이 작다면, 컴파일시에 /N옵션을 사용하여 불필요한 모든 라인번호를 삭제해야 한다. 또한, 변수명을 가능한 한 짧은 단어로 사용해야 한다.

• 어셈블리어의 경우

BIOS 인터럽트 12H는 딥스위치 세팅을 체크하고 AX에 시스템 킬로바이트의 수를 리턴시킨다. 이 값은 8255레지스터 내의 세팅으로 부터 계산된다(리스트 15 참조).

AT는 확장된 메모리를 체크하기 위해 INT 15H의 기능 88H를 사용하며, 이것은 CPU의 어드레스범위가 리얼 어드레스 모드일 때 이 범위 이외의 메모리가 어떤 것인가를 찾아낸다. 즉, 그것은 1M바이트표시 이상의 메모리를 찾아 내는 것이다. 이러한 기능을 수행하기 위해서 시스템 보드는 512~640K정도의 용량을 갖추어야 한다. 확장된 메모리의 1K블록의 수는 AX에 리턴된다.

처음 예는 메모리의 최초 10개의 64K세그먼트에서 64K메모리 뱅크의 수를 체크하는 것이다. 만일 메모리 공간의 상위 6개의 뱅크들 내에서테스트하려면, B000:0000(A000:0000도 가능)에서 시작하는 세그먼트내에는 비디오 버퍼가, F000:0000(C000:0000도 가능)에서 시작하는 세그먼트내에는 ROM이 존재한다는 것에 유의하라(리스트 16 참조).

인터럽트

인터럽트는 컴퓨터가 기본 작업을 수행하기 위해 호출하는 이미 구성된 루틴이며 하드웨어와 소프트웨어 인터럽트 모두 존재한다. 하드웨어 인터럽트는 시스템 보드에서나 I/O 채널내의 카드로 부터 하드웨어에 의해 만들어진 것이다. 즉, 이것은 타이머 칩의 펄스나 프린터로부터의 신호에 의해, 키보드상의 키를 누르거나, 또는 여러가지의 다른 루트를 통해 발생될 수 있다.

하드웨어 인터럽트는 소프트웨어의 운영에 따라 조정되지 않는다. 하나의 인터럽트가 호출되면 CPU는 진행하던 모든 작업을 멈추고 그 인터럽트를 수행한 다음 인터럽트가 나온 곳을 택하게 된다. 프로그램에서 인터럽트가 나온 정확한 곳으로 리턴이 가능하게 되면, 바로 그곳의

〔리스트 15〕

```
;---가용 문장의 수를 찾아 DOS로:
        MOV  AH,4AH      ;SETBLOCK 펑션 사용
        MOV  BX,0FFFFH   ;불가능한 얼로케이션 요구
        INT  21H         ;빈 문장의 번호를 BX가 홀드
```

〔리스트 16〕

```
;---각 메모리 뱅크 테스트:
            CLI                 ;하드웨어 인터럽트를  디스에이블
            MOV  AX,CS          ;코드를 위해 세그먼트 값을 얻는다.
            AND  AX.0FFFH       ;상단 4비트를 자름(하단 세그먼트 세트)
            MOV  ES,AX          ;ES에 위치시킴
            MOV  DI,0           ;DI에 64K 뱅크의 번호를 카운트
            MOV  CX,10          ;10 뱅크의 테스트를 반복
            MOV  BL,'X'         ;재배치 바이트로 'X'사용
NEXT:       MOV  DL,ES:[0]      ;샘플 주소를 저장
            MOV  ES:[0],BL      ;샘플 주소에 X를 위치시킴
            MOV  DH,ES:[0]      ;샘플 주소 리드
            MOV  ES:[0],DL      ;원래 값을 재배치
            CMP  DH,'X'         ;쓰여진 것과 대응이 되는가?
            JNE  GO_AHEAD       ;아니면 볼리에 포함되지 않았는가?
            INC  DI             ;메모리 뱅크의 볼리를 증가시킴
GO AHEAD    MOV  AX,ES          ;포인터를 증가시키기 위해 준비
            ADD  AX,1000H       ;주소 64K 이상을 지시한다.
            MOV  ES,AX          ;ES에 백 포인터를 세트
            LOOP NEXT           ;다음 뱅크를 한다.
            STI                 ;인터럽트를 리인에이블
                                ;이제 DI안에 볼리가 있다.
```

어드레스(CS:IP)는 플래그 레지스터에서와 마찬가지로 스택으로 밀어 넣어지게 된다.

그래서 인터럽트 루틴의 메모리내의 어드레스가 CS:IP로 로드되어 루틴에 제어가 발생하게 된다. 인터럽트 루틴은 때때로"인터럽트 처리자"로서 언급된다. 인터럽트 처리자는 항상 IRET 지시로 끝나는데, 이 지시는 인터럽트를 발생시키는 프로세스를 수행하지는 않으며, CS:IP와 플래그대신에 원래의 값으로 바꿔 넣어 컴퓨터가 계속 작업을 수행할 수 있게 한다.

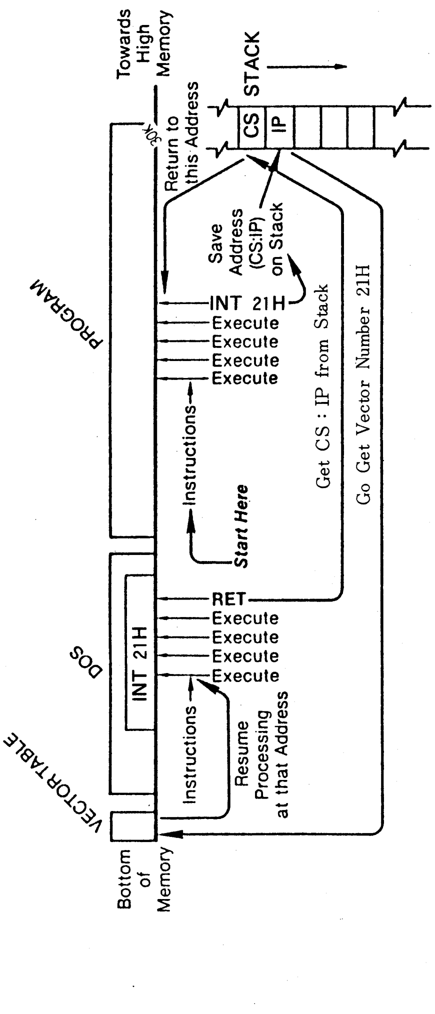
이와는 다르게 소프트웨어 인터럽트는 실제로 어떤 것을 인터럽트하지는 않는다. 이 인터럽트는 근본적으로 단지 실행 프로그램이 키 스트로크(Key Stroke)를 취하거나 스크린상에 라이트하는 것과 같은 세속적인 작업을 수행하기 위해 호출하는 프로시듀어에 불과하다. 그러나, 이러한 루틴은 실행 프로그램이 아닌 운영 시스템내에 라이트되지만, 인터럽트 메카니즘은 프로그램에서 얻을 수 있는 수단이다.

소프트웨어 인터럽트들은 내부적으로 서로 결합될 수도 있다. 예를 들어, 모든 DOS 키보드-입력 인터럽트들은 키보드 버퍼로 부터 캐릭터를 얻기 위해 BIOS 키보드-입력 인터럽트를 사용한다.

하드웨어 인터럽트는 소프트웨어 인터럽트하는 동안 제어를 취할 수 있다는데 유의하라. 이렇게 하여도 어떤 혼란이 일어나지 않는데, 그 이유는 각 인터럽트 루틴들이 변경시키는 모든 레지스터들을 세이브하도록 주의깊게 설계되어 있으며, 최종적으로는 레지스터들을 복원하여 인터럽트가 끝났을 때 CPU에 점유되었던 자취를 남기지 않고 떠나기 때문이다.

인터럽트의 어드레스는 벡터라 부른다. 각 벡터는 4바이트 길이이다. 첫번째 워드는 IP에 대한 값을 유지하며, 두번째 것은 CS에 대한 값을 유지한다. 메모리의 최저1024바이트는 인터럽트 벡터를 갖게 되어 결국 총 256벡터들을 위한 공간이 있게 된다. 이러한 벡터들이 한꺼번에 취해질 때는 벡터 테이블로 언급된다. INT 0에 대한 벡터는 0000:0000에서, INT 1은 0000:0004에서, INT 2는 0000:0008에서 시작된다. 만일 INT8H(요일 및 시간의 인터럽트)에 대한 벡터를 유지하고, 0000:0020에서 시작하는 4바이트를 찾아낸다면, A5FE00F0를 얻게 된다. 각 워드의 하위 바이트는 처음에 메모리로 들어 오고 그 순서는 IP:CS이며, 4바이트 값은 F000:FEA5로 전환된다. 그림2는 INT 21H를 수행하는데 프로그램이 취하는 패스(Path)를 보여준다.

〔그림 2〕 INT 21H의 수행경로



1. 8259 인터럽트 콘트롤러

인텔 8259 프로그래머블 인터럽트 콘트롤러 칩은 하드웨어 인터럽트를 하기 위하여 모든 IBM 마이크로컴퓨터내에 사용되고 있다. 인터럽트에 대한 하나 이상의 요구가 동시에 수행될 수 있기 때문에 칩은 우선순위 배열(Priority Arrangement)을 갖는다. 16레벨의 우선순위를 갖는 AT만을 제외하고는 보통 8레벨을 갖고 있으며, 그 레벨들을 "인터럽트 스탠드(Interrupt Stand)"를 나타내는 약어 IRQ0에서 IRQ7(IRQ0에서 IRQ15)로 부른다. 가장 높은 우선순위는 레벨 0로 간다.

AT에서 여분의 8개 레벨은 제2의 8259칩에 의해 조정되는데, 이러한 제2의 레벨들은 IRQ2와 IRQ3사이에 우선순위를 갖는다. 인터럽트 요구0-7은 INT 8H-INT FH에 대한 벡터들과 대등되지만, AT에서 인터럽트 요구8-15는 INT 70H-INT 77H에 의해 서비스된다. 다음 표 5에 인터럽트 할당을 나타낸다.

요일 및 시간의 인터럽트는 요일 및 시간이 주기적으로 나타났다가 사라지기 때문에 가장 높은 우선순위를 갖는다. 키보드 인터럽트는 키를 누르거나 놓았을 때 발생되며, 이것은 키보드 버퍼(소프트웨어 인터럽트에 의해 복원된 것으로 부터)내에 위치하고 있는 키 코드로 끝나는 연속적인 상황이 발생된다.

〔표 5〕 인터럽트할당

IRQ0	타이머
1	키보드
2	I/O 채널
8	리얼 타임 클럭(AT)
9	소프트웨어를 IRQ2로 바꿈(AT)
10	예약
11	예약
12	예약
13	코프로세서(AT)
14	하드디스크 콘트롤러(AT)
15	예약
3	COM1(AT에서의 COM2)
4	COM2(AT에서의 COM1)
5	하드디스크(AT에서 LPT2)
6	디스크 콘트롤러
7	LPT1

8259는 8개의 하드웨어 인터럽트 라인들을 제어하거나 모니터하는 3개의 1바이트 레지스터를 갖는다. 인터럽트 요구 레지스터(Interrupt Request Register; IRR)는 일치하는 인터럽트 라인이 요구 신호를 내보낼 때 비트를 1로 변화시킨다. 그 칩은 다른 인터럽트가 진행중인가를 자동적으로 체크한다. 마지막으로 인터럽트를 불러내기 전에 그 레벨의 인터럽트가 현재 허용이 되는지 안되는지를 확인하기 위하여 인터럽트 마스크 레지스터(Interrupt Mask Regiser; IMR)가 체크된다. 통상 프로그래머들은 포트어드레스 21H에서 인터럽트 마스크 레지스터를, 포트어드레스 20H에서는 인터럽트코맨드 레지스터만을 액세스한다.

2. 인터럽트의 허용

하드웨어 인터럽트를 쓸모없게 하는 데는 두가지 이유가 있다. 그 하나는 모든 인터럽트들은 보호되도록 하여 컴퓨터내에서 발생하는 다른 어느 것보다 앞서서 중요 코드가 완전히 수행될 수 있도록 하는 것이다. 예를 들어, 하드웨어 인터럽트 벡터가 바뀌는 동안 중간에 인터럽트가 발생하지 않도록 인터럽트들을 쓸모없게 한다.

두번째 이유로는 단지 선택된 하드웨어 인터럽트만이 모습을 감추게 한다. 이것은 그러한 특수 인터럽트들이 어떤 중요한 시점에 방해를 하기 때문이다.

• 어셈블리어의 경우

궁극적으로 모든 인터럽트의 수행은 플래그 레지스터의 인터럽트 플래그(비트 9)에 따르게 된다. 이 인터럽트 플래그는 0일때 인터럽트 마스크 레지스터가 허용하는 어떠한 인터럽트 요구도 수용하게 된다. 만일 인터럽트 플래그가 1이면 어떠한 하드웨어 인터럽트도 발생하지 않는다. 그 플래그를 1로 세트하기 위해서는 CLI명령을 사용하여 인터럽트를 발생하지 못하도록 한다. 그 플래그를 0으로 클리어시키기 위해서는 STI를 사용하여 인터럽트를 재발생시켜야 한다. 요일 및 시간에 대한 인터럽트는 초당 18.2번이 발생하며, 이 인터럽트가 하드웨어 인터럽트들이 발생하지 않는 동안 하나 이상의 요구가 구성되면, 여분의 요구들은 무시되거나 시간 및 요일 카운트가 이루어진 다음에 수행된다.

컴퓨터는 소프트웨어 인터럽트가 발생했을 때는 자동적으로 하드웨어 인터럽트를 사라지게 하거나 이와는 반대의 일을 수행한다는데 유의하라. 소프트웨어 인터럽트를 라이트할 때, 만일 하드웨어 인터럽트를 수행할 필요가 있을 때에는 STI를 갖는 루틴을 사용해야 한다. 또한, STI 다음에 CLI를 사용해야 하는 것을 잊어버린다면, 키보드로부터의 입력은 차단되기 때문에 컴퓨터가 고장을 일으킬 수 있다는데 유의하라.

특수 하드웨어 인터럽트를 숨기기 위해서는 간단히 적당한 비트 패턴을 인터럽트 마스크 레지스터(Interrupt Mask Register;IMR)의 어드레스인 포트 어드레스 21H에 보내면 된다. AT에서 제 2의 8259 칩(IRQ 8-15)내의 마스크 레지스터는 A1H에 있다. 인터럽트를 마스크하기를 원한다면 그 인터럽트의 번호에 일치하는 비트를 1로 세트한다. 그 레지스터는 단지 라이트만이 가능하다. 다음의 예는 마스크 인터럽트의 구성을 보여준다.

프로그램의 끝에서 레지스터를 0으로 클리어하라. 그렇지 않으면 프로그램이 완료된 후에도 세팅은 계속될 것이다(리스트 17 참조).

〔리스트 17〕

```
;---인터럽트 매스크 레지스터에서 비트 6의 매스크:
        MOV  AL,01000000B        ;비트 6을 매스크로(디스크 인터럽트)
        OUT  21H,AL              ;인터럽트 매스크 레지스터에 전송
         .
         .
        MOV  AL,0                ;프로그램 끝에서 IMR을 지움
        OUT  21H,AL
```

3. 인터럽트의 라이트

이용자 자신의 인터럽트를 라이트하는데는 여러가지 이유가 있다. 첫째 이유는 운영 시스템에 의하여 제공되는 이미 구성된 인터럽트의 대부분은 단지 모든 프로그램에 가능한 공통 프로시듀어에 불과하며, 원한다면 이 라이브러리에 도움을 구할 수 있다. 예를 들어, 이용자 프로그램의 대부분은 스크린상에 수직으로 스트링을 라이트하는 루틴을 사용하게 된다. 그 루틴을 프로시듀어로서 각 프로그램에 링크시키는 것보다는 프로그램 종료후에 메모리에 그대로 상주하는 프로그램으로서 라이트되는 인터럽트로 구성하는 것이 좋다. 그래서 CALL WRITE_VERTI_CALLY 대신에 INT 80H를 갖게 될 것이다(인터럽트를 호출하는 것은 프로시듀어를 호출하는 것보다 약간 늦다는데 유의하라).

인터럽트를 라이트하는 두번째 이유는 어떤 특수 하드웨어 인터럽트를 이용하기 위한 것이다. 이러한 인터럽트들은 컴퓨터 하드웨어내의 어떤 존재에 의하여 자동적으로 발생된다. 어떤 경우에 BIOS는 이러한 인터럽트에 대한 벡터를 전혀 작업을 수행하지 않는 루틴(이것은 단지 IRET 문만을 포함한다)을 지시하도록 초기화시킨다. 이용자는 이용자 자신의 루틴을 라이트할 수 있으며 이 루틴을 지시하도록 인터럽트 벡터를 변경시킬 수 있다. 그래서,하드웨어 인터럽트가 발생할때마다 이용자 루틴은 실행된다. 그와 같은 루틴은 하나는 시간 및 요일 인터럽트로 초당 18.2번 자동적으로 발생된다. 통상 이 인터럽트는 시간 및 요일 클럭을 최신의 정보로 새롭게 하지만 인터럽트에서 처럼 어떤 코드도 추가할 수 있다. 만일 이용자의 코드가 클럭 세팅을 체크하거나 지정된 시간에서 동작을 하게 되면 리얼타임 운영이 가능하게 된다. 다른 이용들은 Ctrl-Break, PrtSc, 에러조건 등에 대한 루틴을 프로그램하는 것이다. 프린터 인터럽트와 커뮤니케이션 인터럽트는 컴퓨터로 하여금 I/O 운영과 다른 프로세싱사이에 빠른 속도로 스위치하도록 한다.

마지막으로 운영시스템 루틴의 하나를 전적으로 대신할 수 있는 인터럽트를 컴퓨터의 필요에 맞게 라이트할 수 있다.

• 어셈블리어의 경우

INT 21H의 기능 25H는 인터럽트 벡터를 특정 어드레스로 세트한다. 이 어드레스는 2워드 길이이다. 상위워드는 세그먼트 값(CS에 대한)을, 하위 워드는 옵셋(Offset)(IP에 대한)을 갖는다. 이용자 루틴의 하나를 지시하도록 벡터를 세트하기 위해서는 간단히 DS의 루틴에 대해서 세그먼트를 DX의 루틴에 옵셋시키면 된다(아래 예의 순서를 따른다). 그다음 AL에 인터럽트 번호를 놓고 그 기능을 호출한다. 어떤 인터럽트 루틴은 보통의 RET 지시보다는 IRET로 끝낸다.

기능 25H는 벡터를 변경시켜 자동적으로 하드웨어 인터럽트를 무력하게 하여, 하드웨어 인터럽트가 중간에 벡터를 이용하려는 위험은 없게 된다(리스트 18 참조).

라이트하려는 어떠한 하드웨어 인터럽트의 끝에 다음의 명령을 삽입한다.

```
MOV  AL, 20H
OUT  20H, AL
```

그라인에 똑같은 번호(20H)는 서로 일치한다. 만일 하드웨어 인터럽트가 이코드로 끝나지 않는다면, 8259 칩은 그의 인 서비스 레지스터(In Service Register)를 클리어하지 않아 방금 수행이 끝난 것보다 하위 레벨의 인터럽트를 발생시키지 못한다. 그 코드를 추가하는 것을 잊어버리면 프로그램이 쉽게 깨지게 되는데, 그 이유는 키보드 인터럽트는 쉽게 차단되며, Ctrl-Alt-Del조차도 쓸모없게 되기 때문이다. 이러한 코드는 시간 및 요일 인터럽트에 코드를 추가하는 INT 1CH와 같이 현존 인터럽트에 연달아 추가되는 인터럽트 벡터에 의해 필요없게 된다.

프로그램이 완료됐을때 원래의 인터럽트 벡터는 복원되어야 한다. 그렇지 않으면, 연속되는 프로그램은 그 인터럽트를 호출하게 되어 메모리내에 루틴이 더 이상 존재하지 않는 곳으로 건너 뛰게 된다.

INT 21H의 기능 35는 벡터의 현재 값을 리턴하고, ES에 세그먼트를, BX에 옵셋(Offset)을 위치시킨다. 자신의 인터럽트를 구성하기 전에 이 기능을 사용하여 현재의 값을 얻고 그 값을 세이브한후 프로그램이 완료되기 바로 전에 기능 25H(그 이상으로서)를 사용하여 그 값을 복원한다. 예를 들면 리스트 19와 같다.

인터럽트를 라이트할 때 조심해야 할 한 쌍의 함정(pitfall)이 있다. 만

〔리스트 18〕

```
;---인터럽트를 셋업:
            PUSH DS                 ;DS를 저장
            MOV  DX,OFFSET ROUTINE  ;DX에서 인터럽트 루틴의 오프셋
            MOV  AX,SEG ROUTINE     ;인터럽트 루틴의 세그먼트
            MOV  DS,AX              ;DS에 위치
            MOV  AH,25H             ;벡터를 셋업하기 위한 펑션
            MOV  AL,60H             ;벡터번호(INT 60H)
            INT  21H                ;인터럽트를 변화시킴
            POP DS                  ;DS를 저장

;---인터럽트 루틴:
 ROUTINE    PROC FAR
            PUSH AX                 ;모든 변화된 레지스터를 저장
             .
             .
             .
            POP  AX                 ;모든 변화된 레지스터를 재저장
            MOV  AL,20H             ;하드웨어 인터럽트를 위해
            OUT  20H,AL             ;이 두라인을 사용한다.
            IRET
 ROUTINE    ENDP
```

〔리스트 19〕

```
;---데이터 세그먼트에서:
      KEEP_CS     DW   0      ;재배치 인터럽트를 위해 세그먼트를 홀드
      KEEP_IP     DW   0      ;재배치 인터럽트를 위해 오프셋을 홀드

;---프로그램의 처음에서:
            MOV  AH,25H     ;INT 주소를 얻기위한 펑션번호
            MOV  AL,1CH     ;벡터의 번호(INT 시계)
            INT  21H        ;현재 세그먼트는 ES에 오프셋은 BX에 있다.
            MOV  KEEP_IP,BX ;오프셋 저장
            MOV  KEEP_CS,ES ;세그먼트 저장

;---프로그램의 끝에서:
            CLI
            PUSH DS         ;DS는 파괴되었다.
            MOV  DX,KEEP_IP ;오프셋을 재저장하기 위해 준비
            MOV  AX,KEEP_CS ;
            MOV  DS,AX      ;세그먼트 재저장하기 위해 준비
            MOV  AH,25H     ;펑션을 인터럽트 벡터에 세트시킴
            MOV  AL,1CH     ;벡터의 번호
            INT  21H        ;인터럽트가 리세트되었다.
            POP  DS         ;DS를 재 저장
            STI
```

일 새로운 인터럽트 루틴이 그 내부에 데이터 변수들을 액세스(access)할 필요가 있다면 DS가 적당히 세트(set)되었는 지를 유의하라(통상 인터럽트는 호출 프로그램에 의해 제공되는 스택을 사용할 수 있다). 또 다른 고려는 Ctrl-Break를 통해 끝나는 프로그램이 Ctrl-Break 인터럽트 그 자체가 잡(job)이 수행한 인터럽트를 찾도록 프로그램되지 않는 한 변하는 인터럽트 벡터를 복원하는데 실패하는가에 대한 것이다.

위에서 설명한 DOS 기능들은 단지 메모리의 하위 끝의 2워드를 복원하거나 변경시키는 것이다. 벡터의 옵셋은 간단히 그 벡터의 번호에 4를 곱함으로서 얻어진다.

즉, INT 16H의 어드레스를 ES:BX내에 놓는다. 예를 들면 리스트 20과 같다.

DOS기능으로 우회하거나, 인터럽트 벡터를 직접 세트하는 것은 바람직하지 않다. 특히 멀티태스킹 환경하에서 운영 시스템은 다수의 인터럽트 벡터 테이블을 지원할 수도 있으며, 그리고 그 테이블의 실제 배치는 단지 DOS에만 알려질 수 있다.

4. 인터럽트의 할용

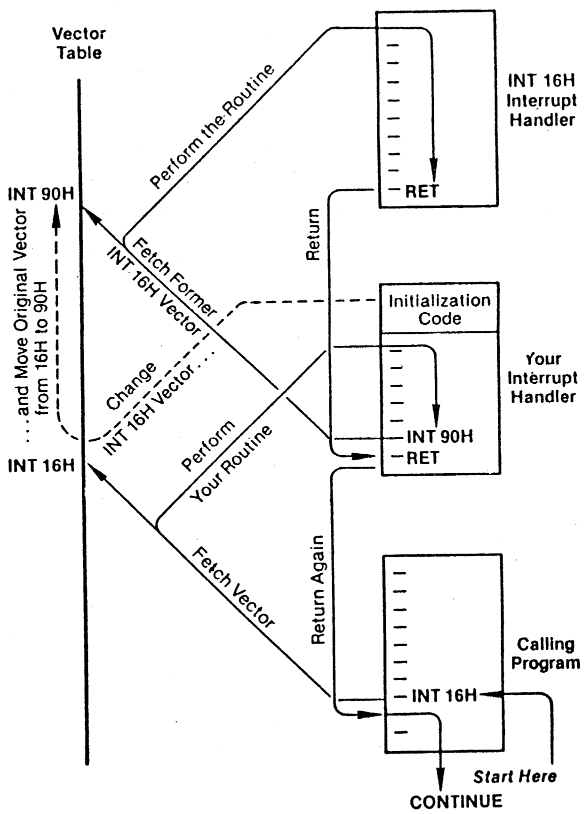
비록 보편적인 것은 아니지만, 때때로 현존 인터럽트에 형태를 추가하는 것이 유용하다. 예로서 하나의 키를 침으로써 캐릭터들의 긴 이용자정의 스트링으로 전환하는 소프트웨어 유틸리티를 고려해 보자. 이러한 유틸리티는 BIOS 인터럽트 16H의 기능 0를 통해서 모든 표준 키보드 입력이 도착한다는 사실을 이용할 수 있다. 기본 DOS 키보드입력 인터럽트는 키보드 버퍼로부터 캐릭터를 취하기 위해 BIOS 인터럽트를 호출한다. 그래서 필요한 것은 단지 INT 16H가 매크로에 대한 게이트웨이(GateWay)로서 작용하도록 수정하는 것이며, 그렇게 되면 어떠한 프로그램도 키보드입력 인터럽트를 사용하든지 안하든지 간에 매크로를 받아들이게 된다.(정교한 키보드 매크로 프로그램은 키보드 인터럽트를 INT 9로 완전히 교체한다.)

이때 단지 BIOS 인터럽트만이 수

〔리스트 20〕

```
;---INT 16H의 주소 얻기:
        SUB  AX,AX          ;메모리의 최하단에 나머지 세그멘트를 세트
        MOV  ES,AX          ;
        MOV  DI,16H         ;DI에 INT 16H를 준다.
        SHL  DI,1           ;*2
        SHL  DI,1           ;다시 *2
        MOV  BX,ES:[DI]     ;BX에 하위바이트를 준다.
        MOV  AX,ES:[DI]+2   ;ES에 상위바이트를 준다.
        MOV  ES,AX          ;
```

〔그림 3〕 인터럽트의 수행결과



정되어 진다. DOS 인터럽트는 다시 들어가지 않는다. 또한, BIOS 인터럽트는 ROM에 있기 때문에 내부적으로 수정될 수는 없다. 그러나, BIOS 인터럽트에 앞서거나 뒤따르는 서브루틴을 라이트할 수 있으며, 이러한 서브루틴은 인터럽트가 호출될 때마다 발생된다. 예를 들어, INT 16H의 경우에 루틴을 단지 라이트하여 이 루틴에 INT 16H에 대한 인터럽트 벡터를 자지한다. 어떤 비사용 벡터로 전달된 INT 16H에 대한 원래의 벡터를 60H 인터럽트를 이용하기 위해 간단히 INT 60H를 호출한다. 즉, 프로그램이 INT 16H를 호출하였을 때, 제어는 특수 루틴으로 전달되며, 이 루틴은 원래의 16H 인터럽트를 호출하게 되고, 이 인터럽트가 수행이 끝났을때 특수 루틴으로 되돌려서, 새로운 루틴이 프로그램내에 16H에 대한 호출이 이루어진 곳으로 리턴된다. 이것이 일단 이루어지면 INT 60H가 호출되기 전이나 후에 특수 코딩이 새로운 루틴내에 놓일 수 있게 된다. 그림 3은 이러한 과정을 보여준다. 요약하면

1. INT 60H를 호출하는 어떤 시점에서 새로운 루틴을 구성하라.
2. 16H에 대한 벡터를 60H로 전달하라.
3. 새로운 루틴을 지시하도록 16H 벡터를 변경하라.
4. 메모리내에 프로그램을 남기면서 프로그램을 종료하라.

과 같이 수행됨을 알 수 있다.

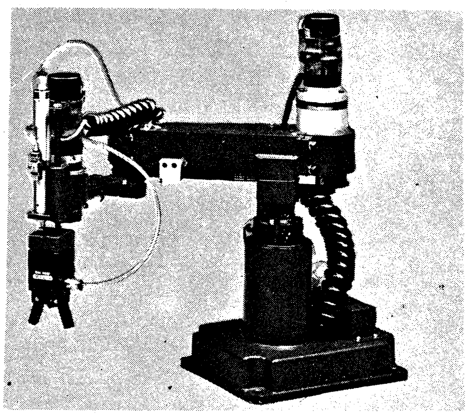
마이크로프로세서를 이용한

FA SYSTEM 과정

(생산자동화)

※ 개강 :
매월 5일

講義案內 (2개월 완성)

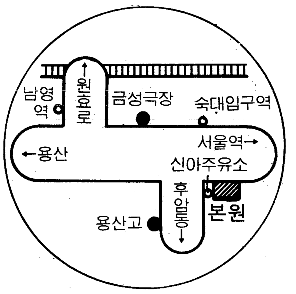


본원에서는 다년간 Computer Hardware 과정을 교육하여 많은 수료생을 배출, 각계에서 공헌하고 있는 바 금번 생산업계의 요청에 따라 컴퓨터 프로세서를 이용한 생산체제와 산업용 로보트의 발전으로 제품과 제조공정의 설계 및 계획, 공장 현장에 있는 생산 제조장치의 컴퓨터 제어, 원료 및 자재의 유동제어, 공간 활용 및 작업계획 등의 조직적 기획 등의 필요성에 따른 FA 과정의 이론, 실기를 위주로 특강을 하게 되었으니 관계 직원의 많은 참여 바랍니다.

■ **강의일정**

주별 \ 일자 (과정)		강습내용: 이론및실기	주별 \ 일자 (과정)		강습내용: 이론및실기
1주	1일차	Logic회로 설계의 기초	5주	1일차	PWM System
	2일차	Micom System		2일차	Power Control System
	3일차	KKH 80A H/W구조		3일차	DC Motor Control
2주	1일차	Z80 Assembly 언어	6주	1일차	Step Motor의 기초 이론
	2일차	KKH 80A S/W 구조		2일차	Step Motor Control
	3일차	PIO 입출력 실습		3일차	Senser Interfacing(1)
3주	1일차	A/D Converter의 제작	7주	1일차	IBM PC H/W 구조
	2일차	D/A Converter의 제작		2일차	IBM PC/XT Interfacing
	3일차	DC Motor Drive회로의 설계		3일차	RS 232C Serial Interfacing
4주	1일차	Monitor Program의 설계	8주	1일차	Senser Interfacing(2)
	2일차	응용 Program 설계(1)		2일차	Micro Robot(1)
	3일차	응용 Program 설계(2)		3일차	Micro Robot(2)

전자산업 4반세기를 지켜온 기술인의 전당



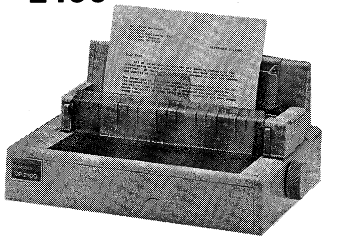
과인 **한국전자통신학원** 컴퓨터 기술부

(구 한국전파학원)

140 서울시 용산구 갈월동 17번지 ☎ 754-1751/2902/1044

신제품

저가격의 보급형 프린터
DP-2100



최경량의 콤팩트한 디자인으로 개발된 이 제품은 한자를 ROM에 내장했으며 저소음 및 그래픽의 고속인쇄, 그리고 사용이 편리한 콘트롤 패널기능과 부품의 모듈화로 인해 유지보수가 용이하다는 장점을 갖고 있다.

접속가능한 컴퓨터로는 대우 PRO 시리즈 전기종과 IBM-PC/XT/AT 호환기종, MSX컴퓨터이며 옵션으로는 낱장공급장치, 전산용지 FEEDER, RS-232C I/F카드등이 있다. DP-2100 의 주요 규격을 보면 다음과 같다.

- 프린트 방식:9핀 도트매트릭스
- 프린트 방향:양방향(최단거리 이동)
- 인쇄속도:영문, 숫자- 150cps(DRAFT), 50cps(NLQ)/한글, 한자-25cps
- 문자체:영문-DRAFT/한글, 한자-고딕체
- 문자구성:영문, 숫자-9×9(DRAFT), 18×9(NLQ)/한글, 한자-16×16, 14×16
- 사용문자:조합형 한글, 신문방송협회 한자 5,170자, KS표준한자 4,888자, 영문, 숫자, 그래픽문자
- 인자기능:밑줄, 굵은문자, 역상, 상하첨자, 가로2배확대, 세로2배확대
- 행당 인자수:영문, 숫자-80자/한글, 한자-53, 68자
- 복사매수:원본포함 2매
- 인터페이스:센트로닉스 패러렐
- 리본:스몰 카트리지(수명:80만자)

제조원 및 판매원 : 대우통신
전 화 : (02)771-35 (교환) 4133/34

다목적용 미니급 컴퓨터
SAM PC-386



SAM PC-386은 주기억용량이 2MB로써 대용량의 데이터처리 능력과 함께 기계설비 및 냉난방 부하계산등 고도의 엔지니어링분야와 디지타이저, X-Y플로터, 마우스, 라이트-펜 등의 주변장치를 이용하여 건축, 기계, 전자, 전기, 토목등 각종 분야의 CAD/CAM 시스템으로 적극 활용할 수 있다.

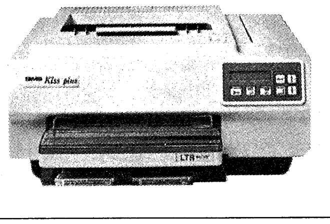
또한 이 시스템은 IBM PC/AT의 모든 영문 S/W와 호환성을 가지며 OS로는 MS-DOS 3.2이상, PC-DOS 3.1이상 및 멀티유저용의 XENIX, 그리고 UNIX도 가능하다. 특히 주기억용량은 최고 25MB까지 확장 가능하고, 40MB를 기본으로 내장하고 있는 HDD는 최고 280MB까지 확장할 수 있어 대량의 데이터를 처리할 수 있는 본격 OA용 컴퓨터이다.

주요 규격을 보면 다음과 같다.

- CPU:i80386, 32비트(16MHz/20MHz)
- Co-Processor:i80387(선택)
- ROM:64KB(표준)
- RAM:2MB(표준)
- 보조기억장치:FDD:5.25″(1.2MB) HDD:40MB
- 인터페이스:패러럴포트, RS-232C 시리얼포트
- 키보드:101키
- 화면해상도:720×348
- 확장슬롯:8개
- OS:OS/2, MS-DOS, XENIX, PC-DOS, CPC-DOS386
- 선택:한글한자 카드, EGA 카드
- 터미널기능:최대 32대 접속가능

제조원 : (주)SAM KOREA
판매원 : 선진전자
전 화 : (02)275-3709

레이저프린터
QMS KISS PLUS



사무자동화의 요구에 부응하여 소형 경량의 탁상형으로 설계된 레이저프린터로 퍼스널컴퓨터용으로 적합하게 제작되었으며, 카드 모델에 따라 그 기능을 확장할 수 있다.

제품의 주요 규격은 다음과 같다.

- 엔진:CANON SX
- 인자방식:반도체 레이저에 의한 건식 전자사진방식
- 해상도:300×300 DPI(Dot Per Inch)
- 인자속도:8PPM(Page Per Minute)
- 듀티사이클:5,000페이지 / 월
- 토너 카트리지 수명:4,000페이지 / 월
- 시동시간:60초
- 사용용지:A4(210×297㎜), LETTER(216×279㎜), LEGAL(216×356㎜)
- 영문폰트:17
- 인자방향: Portrait, Landscape
- 인터페이스:RS232C / 센트로닉스 겸용
- 메모리:DRAM－1MB / 2.5MB
USER MEMORY－750KB / 2.1MB
- 크기:457㎜(W)×231㎜(H)×635㎜(D)
- 중량:20KG
- 전원:110 / 120V 60Hz, 220 / 240V 50Hz, 1KW(최고)
- 환경:온도:10～32도
습도:20～80%RH
소음:55dB 이하

제조원 및 판매원 : 대우통신
전 화 : (02)771－35(교)4133

다이얼업 모뎀
X-TAL 24D

X－TAL 24D는 2천4백bps급의 전송속도를 지니며 EIA RS－232C / CCITT V.24 / V.28 표준 규정에 맞춰 설계된 것으로 자동다이얼링 및 에러 자동복구 기능을 지닌 제품으로서 주

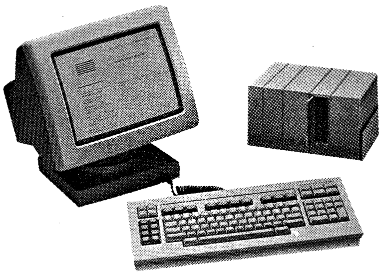


요 규격은 다음과 같다.

- 적용:동기, 비동기식 이진 시리얼 데이터 전송
- 전송속도:동기식－2400, 1200 / 비동기식－2400, 1200
- 물리적 인터페이스:RS-232C(25핀)
- 디지탈 인터페이스:EIA RS-232-C / CCITT V.24 / V.28
- 사용 라인:전화네트워크 라인
- 라인 및 전화망 접속:RJ－11 폰잭
- 모듈러 설계:Bell 103 / 113,. 212A, Bell 2400, V.22A, V.22B
- 전송VF인터페이스:키보드를 통한 -9dBm～-23dBm사이의 제어신호
- 수신VF인터페이스:0～48dBm의 동작범위
- 제어모드:매뉴얼 및 노르말 모드
- 전송 데이터버퍼:1K 캐릭터
- 흐름제어:CTS 또는 X－on / X－off
- 전원:110 / 230 VAC±10%
- 전력:5W(최대)
- 주파수:60Hz

제조원 및 판매원 : (주)콤텍시스템
전 화 : (02)715－4911

인텔리전트 단말기로 적합한
NGEN 286i



NGEN시스템은 강력한 온라인 단말기능으로 메인 프레임의 방대한 정보를 신속하게 이용할 수 있으며 인텔의 80286, 80386 CPU의 장점을 고려하여 설계된 CTOS / VM을 OS로 채택, LAN기능의 제공과 서로 다른 구성이나 프로세서 또는 응용 소프트웨어를 운영하는 WORK GROUP을 상호 접속시켜 토탈네트워크 시스템을 구축할 수 있다. NGEN 286i 주요 규격은 다음과 같다.

- CPU:i80286(8MHz)
- CO－Processor:i80287(8MHz)
- ROM:16KB
- RAM:1MB(표준)(최고 2GB)
- 보조기억장치:FDD:5.25″(1.2MB)
HDD:20MB(최고 2GB)
- 최대 드라이브수:7개
- I / O 인터페이스:시리얼－RS－232C ×2 / RS－422(1.8M비트 / sec), 패러럴－센트로닉스 규격
- 내부 / 외부확장슬롯(X-BUS):2개
- 전원:AC 115 / 230V, 60Hz
- 소비전력:200W
- 키보드:98키
- 화면해상도:1024×768

제조원 및 판매원 : (주)한독
전 화 : (02)540－4381～8

CAD용 플로터
HP 드래프트 EXL, DXL



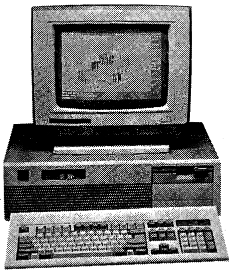
HP의 이 제품은 PC CAD시장을 겨냥한 것으로 각각 A사이즈(8×11인치)에서 E사이즈(34×44인치), A에서 D사이즈(22×34인치) 정도의 도면작업이 가능하다. 따라서 중형 플로터의 선택폭을 넓혔을 뿐만 아니라 건축, 공학,

건설등에 이용되는 기계식 설계나 전자식 설계에 적합하며 대부분의 PC기종과 공학워크스테이션 CAD시스템과 연결이 가능하다.

또한 동제품은 펜식별 능력을 갖추고 있어 펜을 바꾸는 횟수를 최대한 줄임으로써 설계시간을 단축시켰고, 8개의 회전펜은 자동적으로 뚜껑이 덮여져 잉크가 마르거나 펜촉이 상하는 위험을 방지하고 있다.

제조원 : HP
판매원 : 삼성HP
전 화 : (02) 784 - 2666 (교) 419

26.7MHz 클럭속도의 컴퓨터
트라이던트 AT



미국 실리콘벨리에서 제작된 제품으로 Chips & Technology사의 VLSI LOGIC의 NEAT칩을 장착 IBM PC/AT와 훌륭한 호환성을 유지하며 20MHz의 클럭속도에 Zero Wait에서 26.7MHz로 동작해 Autocad, Versacad, Orcad, Prodesign 등 CAD/CAM/CAE 분야에 적합한 시스템이다.

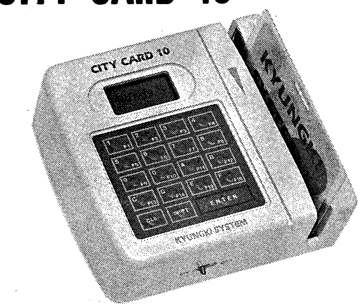
또한 한글보드와 S/W를 지원해 한글 WP, 스프레드 시트, 디베이스등 국내 OA업무에도 어울리게 되었다.

이 제품의 주요 규격은 다음과 같다.

- CPU:Intel 80286(20MHz)
- ROM:1MB(최고 16MB)
- 램의 액세스속도:80nS
- 보조기억장치:FDD:1.2MB
  HDD:44MB
- 디스플레이 카드:한글, 영문동시처리 HGC 또는 EGA 자동절환.(MGA, CGA, EGA)
- 리얼타임클럭 내장
- OS:MS-DOS, OS/2, SCO XENIX, NOVELL-NETWARE
- 키보드:101키

수입원 : (주)설남
판매원 : (주)설남
전 화 : (02)701 - 3293～4

신용카드 조회 전용
CITY CARD-10



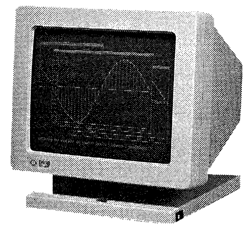
CITY CARD-10은 은행, 유통업계, 증권회사 등 신용카드를 조회해야 하는 산업분야에 사용되는 마그네틱 카드리더로서 사용이 정지된 정지카드와 사용카드의 구분가능하며 호스트 컴퓨터에서 수신한 사용정지카드도 검출가능하다. 또한 온라인과 오프라인을 선택하여 사용할 수 있으며 조회시간이 카드 1장당 2초 이내이며 신속한 처리와 기존 전화선을 이용할 수 있으므로 설치가 매우 간단하다.

주요 규격을 보면 다음과 같다.

- 형태 : 마그네틱 카드 리더
- 통신방식 : RS-422A
- 통신거리 : 1.2km
- 통신속도 : 4800～9600bps
- 표시장치 : LCD 8문자 1라인 버저
- 키패드 : 19키
- 전원 어댑터, 입력 : AC110/220V 60Hz, 출력 : 9V
- 외형 : 116×107×35(mm)
- 중량 : 260g

제조원 및 판매원 : 경기시스템(주)
전 화 : (02)585 - 7781

14인치 평면 모니터
MM-88F



이 모니터는 IBM PC계열과 호환성을 가지도록 제작되었으며 GREEN, PAPER WHITE, AMBER 등 다양한 CRT색상을 사용자가 선택할 수 있다.

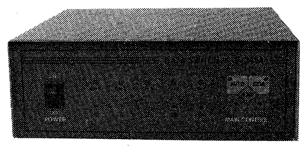
또한, 인체공학적인 설계(회전 받침대)와 SMPS를 사용하여 100V～240V AC FREE VOLTAGE방식을 채택하고 있다.

주요 규격을 보면 다음과 같다.

- CRT : 14" FLAT SCREEN TUBE
- 입력신호 : TTL
- 스캐닝 주파수 : H. F—18432kHz
  V. F—50Hz
- 비디오 RISE/FALL시간:최고 20ns
- 글자수 : 80×25
- EXT. 콘트롤 : BRIGHTNESS, CONTRAST
- 디스플레이 영역 : 230×170mm
- CRT색상 : GREEN, PAPER WHITE, AMBER
- 전원 : 100～240V(FREE VOLTAGE SMPS)
- 동작 환경 : 온도 5～40℃
  습도 5～90%
- 보관 환경 : 온도 －10～55℃
  습도 5～90%

판매원 : 매그넘전자(주)
전 화 : (02)552－7279

전자식 데이터스위치
KP-345A



이 제품은 여러대의 PC가 한대의 프린터를 공동으로 사용할 수 있게 함으로써 경비를 절약할 수 있게 개발되었다.

입·출력포트는 센트로닉스 규격에 준하여 사용하며 시간조정 DIP S/W를 통하여 컴퓨터와 컴퓨터를 절환할 때 대기시간을 5~40초까지 조정할 수 있다.

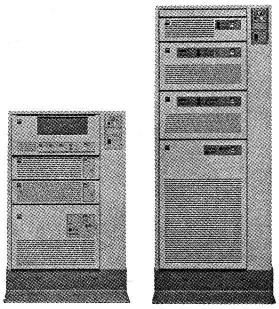
또한 시스템의 상태를 표시하는 LED 기능을 구비하고 있는데 전원 ON시 완전자동의 AUTO상태가 처음 설정되며 사용자가 반자동으로 변경할 수도 있다.

그밖의 주요 특징과 규격을 보면 다음과 같다.

- 입·출력방식 : 센트로닉스 규격
- 입력(컴퓨터) : 2대~8대(최고 64대)
- 출력(프린터) : 1대
- 채널선택 방식 : 자동, 반자동
- 모니터용 LED : 7~19EA
- 사용전압 : AC 100V±10%
- 소비전력 : 30W 이하
- F.F삭제기능 : 용지 F.F기능이 ON시 중복, F.F코드는 자동 소멸
- LED기능 : 사용중인 PC 및 신호의 입력여부 확인
- 채널당 데이터 스위치 가격 : 기존 프린터의 1/10~1/30

제조원 및 판매원 : ATTA SYSTEM
전 화 : (02)277-1739

IBM 9370 정보시스템
ES/9370시리즈



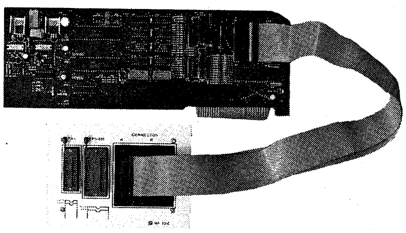
9373 모델 20, 9375 모델 40, 60, 9377 모델 90등의 프로세서를 갖춘 이 시스템은 IBM시스템 9370 아키텍처 가운데 가장 작은 기종으로 복잡한 수치계산을 요하는 과학기술계와 분산처리 실현 및 비IBM기종과의 접속을 구현하는데 적합하다. 또한 이 시스템은 독자적으로 호스트 또는 분산처리용 시스템으로 사용할 수 있다.

주요 제품 규격은 다음과 같다.

제조원 : IBM
판매원 : 한국IBM
전 화 : (02)781-6941

프로세서	9373	9375	9377
메모리	4~16MB	8~16MB	8~16MB
9332/9335 DASD	368MB~6.6GB	368MB~13.2GB	368MB~39.6GB
시스템/370 채널	0~1	0~2	0~12
Aggregate I/O Capacity	5.5MB/sec	22MB/sec	39MB/sec
통신 콘트롤러	2	4	12
워크스테이션 콘트롤러	2	6	12
DASD/테입 콘트롤러	2	4	12
I/O 슬롯(최대)	7	17	54

PAL/EPROM 프로그래머
SYS-K005



이 제품은 시스템을 개발할 때 꼭 필요한 PAL과 ROM의 프로그래머 기능뿐만 아니라 여러가지 유용한 소프트웨어도 제공되고 있어서 개발효율을 높일 수 있다.

특히 PAL기능의 MMI PAL 어셈블러 화일 PAL 20, PAL 24에서 생성되는 BRF 화일을 로드해서 라이트하는 기능은 주목할만 하다.

PAL기능과 EPROM기능의 특징들을 살펴보면 아래와 같다.

■ **PAL기능**

- MMI, NS, TI사의 20핀, 24핀 PAL 타입 라이트 가능
- 사용이 편리한 오퍼레이팅 소프트웨어
- 소프트웨어 : Write, Read, Protect, Edit, Print verify, Load ASM
- PAL 어셈블러에서 생성된 BRF 화일을 로드해서 라이트하거나 지원 소프트웨어에서 PAL 프로그램을 편집해서 직접 라이트 가능
- 프로텍트기능으로 하드웨어의 노하우 보호

■ **EPROM기능**

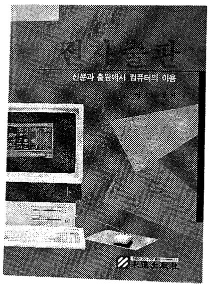
- 각종 화일을 직접 ROM에다 라이트 가능
- 2716~27512 EPROM타입 프로그램 가능
- 편리한 오퍼레이팅 소프트웨어
- 라이트시 시간 : 2716 : 6초, 27256 : 9초, 2732 : 12초, 27512 : 192초, 27128 : 48초
- 소프트웨어 : 라이트, 리드, 베리파이, 화일 저장, 공백 체크, 복사, 로드 HEX(로드 HEX는 X-어셈블러에서 생성된 HEX코드를 로드시켜 주어 라이트시 HEX코드를 라이트할 수 있다.)

제조원 및 판매원 :
SYSTEK ENGINEERING
전 화 : (02)587-5469, 581-8085

신간안내

전자출판

이기성 저/영진출판사 발행/
정가 8,000원/546면/4·6배판



모든 분야에서 컴퓨터의 이용이 활발해지고 필수요건으로 되고 있는 현실에서 본서는 컴퓨터와 신문, 출판업무를 연결시키는 과정을 이론적으로 정립시킨 출판업무 전산화를 위한 지침서라 할 수 있겠다.

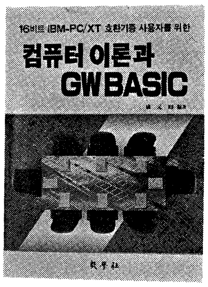
책의 서두에는 컴퓨터와 그에 사용되는 프로그램 언어에 대하여 개괄적으로 설명하고 또한 전자출판에서 이용되는 워드프로세서와 데이터베이스에 대한 기초적인 사용법 등도 설명하였다.

후미에는 출판에서 통신의 이용－전자우편, 한글전자 사서함－형태와 전자출판에 대해서도 자세히 서술하고 있다.

필자가 여러 대학등에서 강의하던 내용을 모아 엮은 것으로 국내 최초의 전자출판 이론과 응용서로서 독자들에게 많은 도움이 되리라 생각한다.

컴퓨터 이론과 GWBASIC

강원경 편저/교학사 발행/
정가 5,500원/373면/4·6배판



얼마전까지만 해도 컴퓨터라 하면 8비트 퍼스널컴퓨터가 우리 생활의 대부분에 이용되어 왔다. 그러나 지금은 16비트 퍼스널컴퓨터를 사용하는 시대로 바뀌고 있다.

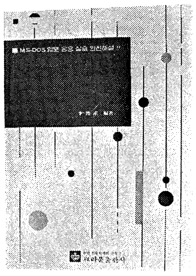
따라서 이미 8비트용 컴퓨터를 배워 잘 다룰줄 아는 사람도 16비트용 퍼스널컴퓨터를 조작하기 위해서는 체계적인 공부를 할 필요성이 있다.

본서는 애플이나 Z80의 8비트 퍼스널컴퓨터에서 16비트 퍼스널컴퓨터로 전환되는 과정에서 반드시 알아야 할 컴퓨터의 기초이론과 가장 보편적으로 널리 쓰이고 있는 언어인 GWBASIC의 명령어들을 소개하고 있다.

그리하여 본서는 초보자나 컴퓨터에 관심이 있는 사람, 특히 혼자서 공부하기를 원하는 사람들을 위해 IBM-PC/XT 호환기종을 대상으로 함축성 있으면서도 집약적인 형태로 컴퓨터에 관한 이론을 전개하고 있다.

MS-DOS 핸드북

이준구 편저/크라운출판사 발행/
정가 9,000원/775면/4·6배판



16비트 퍼스널컴퓨터의 보급과 아울러 마이크로소프트사의 MS-DOS 역시 세계적으로 가장 많이 사용되는 소프트웨어로 부상하고 있다.

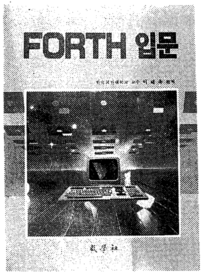
본서에서는 마이크로소프트사의 MS-DOS의 각 버전 발전사와 컴퓨터를 처음 다루는 사람을 위한 기본적인 작동법과 하드웨어 관련부분 즉, 주변장치 접속을 위한 부분과 DOS의 디바이스를 다루어 DNS접속, 통신망접속, 프린터접속시에 도움이 되리라 본다.

또한 17장, 18장에서는 마이크로소프트사의 걸작이라 할 수 있는 Window에 대하여 설치, 운영하는 방법을 설명하였다. 부록에는 DOS의 참고사항, ASCII 코드표, DOS의 각종 메시지와 DOS를 업그레이드하는 법을 자세히 다루고 있다.

컴퓨터 사용시 필요로 하는 모든 지식을 담고 있어 초보자들에게는 훌륭한 지침서가 되리라 생각한다.

FORTH 입문

이태욱 편저/교학사 발행/
정가 4,000원/118면/4·6배판



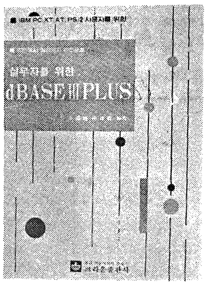
제 4세대 언어의 주역이 될 FORTH는 찰스 무어(Chalse Moore)가 제작한 언어로서 사용자가 직접 핵심되는 단어들을 이용해서 수많은 단어들을 정의해서 쓸 수 있는 언어이다.

본서는 FORTH를 소개하고 FORTH의 기초부분을 중심적으로 설명하였으며 FORTH의 주변환경에서 뒷받침 해주는 개념까지도 개괄적으로 열거하였다.

한편 제7장에서 FORTH 연습 100제를 실어 독자의 이해를 돕고 있다.

실무자를 위한
dBASE Ⅲ PLUS

전용진·신제훈 편저/
크라운출판사 발행/
정가 6,000원/353면/4·6배판



사회가 다변화됨에 따라 오늘날에는 인간이 처리하고 기억해야 할 자료가 상당히 방대해지고 있는 실정이다. 그러므로 이러한 자료를 효율적으로 처리하

고 관리하기 위한 도구가 필요하게 되었다.

이에 본서에서는 컴퓨터를 이용하여 자료를 관리할 수 있는 Ashton-Tate사에서 개발한 dBASE Ⅲ PLUS를 편역하여 자세히 설명하고 있다.

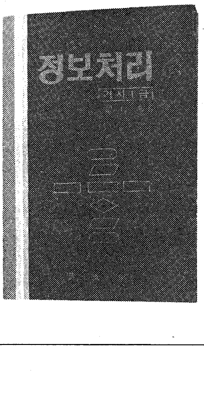
dBASE Ⅲ PLUS는 개인 및 기업의 자료저장, 검색문제를 효율적으로 해결해 줄 수 있는 데이터베이스관리용 프로그램으로 퍼스널컴퓨터상에서 가장 많이 사용되고 있다. 특히 dBASE Ⅲ PLUS는 메뉴방식과 코맨드방식 모두를 제공하기 때문에 이용자가 선택하여 사용할 수 있다.

본서는 퍼스널컴퓨터의 표준 OS(운영시스템)로 채택되어 있는 MS-DOS나 PC-DOS하에서 dBASE Ⅲ PLUS 사용방법을 각 주제별로 쉽게 서술하고 있다.

dBASE Ⅲ PLUS를 이용하여 자료를 관리하는 이용자에게는 많은 도움이 될 것이다.

기사 1급 **정보처리**

나인화 저/청문각 발행/
정가 6,500원/529면/국판



정보산업의 발전에 따라 자료처리 분야가 크게는 기업경영에서부터 일상생활에 이르기까지 컴퓨터에 대한 관심이 날로 광범위하게 늘어나고 있는 실정이다.

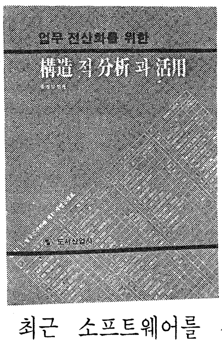
이에 따라 정보산업 분야의 전문적인 인력양성과 함께 체계적인 전문요원 관리육성을 위한 국가자격이 요구되고 있다.

본서에서는 종전의 체계를 개정하여 새 경향에 따른 철저한 문제분석과 일본의 최신 정보처리문제를 분석 검토하여 완벽한 방향제시와 더불어 범위를 압축할 수 있도록 하였다.

또한 기초지식이 다소 부족한 독자들도 쉽게 이해할 수 있도록 문제들을 상세히 풀이하였고 어떤 유형의 문제를 접하더라도 응용력을 배양할 수 있게 하였다.

업무 전산화를 위한
구조적 분석과 활용

유정일 편저/도서산업사 발행/
정가 6,000원/333면/4·6배판



최근 소프트웨어를 산업분야의 하나로 독립, 정착, 육성시키기 위한 시책으로「소프트웨어 개발촉진법」이 제정되기에 이르렀다. 이러한 사회적인 여건에 부흥하여 구조적 분석을 다룬 본서는 인력수급상의 문제, 소프트웨어의 질적향상, 소프트웨어의 생산성향상 등에 기여할 수 있으리라 본다.

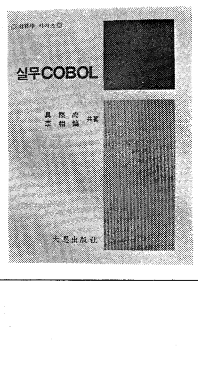
구조적 분석의 특징은 아무리 거대한 업무라 할지라도 이를 작게 세분하여 분석해 나가므로써 누구나 쉽게 파악하고 이해할 수 있으며, 또한 구조적 기법과 도구를 이용하여 문서화하므로써 사용자와의 의사전달이 명확하고 용이하도록 하여 사용자의 요구사항이 무엇인지를 바르게 파악함으로써 이상적인 전산화시스템 구현이 가능하다.

따라서 전산에 대한 기초지식이 없는 사람이라 할지라도 이 기법과 도구를 익히고 활용하면 어떠한 업무라도 쉽게 분석하고 설계하여 성공적으로 업무의 전산화를 실현할 수 있으리라 본다.

한편, 본서에서는 구조적 분석 적용사례에 많은 지면을 할애하여 소개하고 있다.

전산학시리즈
실무 COBOL

구연설·이상호 공저/대은출판사 발행/
정가 7,000원/408면/4·6배판



본서는 가장 오랜 역사를 가진 전통적 자료처리 언어인 COBOL을 익힌 초급프로그래머가 실무에 쉽게 적응하여 고급프로그래머로서 발돋움할 수 있도록 다음과 같은 점에 역점을 두었다.

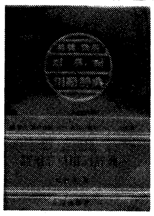
첫째, COBOL 기초과정을 이수한 사람이 COBOL 언어를 효과적으로 구사하여, 그 특징이나 장점을 살릴 수 있도록 다양한 표현방법을 적용한 프로그램 예제를 정선, 수록하였다.

둘째, 모든 프로그램은 구조적 프로그래밍기법을 이용하여 작성함으로써 프로그램개발의 생산성향상과 유지보수가 효과적으로 행해지는 소프트웨어 개발방법을 익히게 하였다.

세째, 기초과정에서 익힌 프로그래밍 기법을 체계화하고 실무에 쉽게 적응할 수 있도록 프로그래밍을 형태별로 분류하여 구성하였다.

네째, 각 단원의 프로그램 예제들은 체계적인 소프트웨어 개발방법에 따라 입출력설계, 처리방법의 제시, 개략 순서도, 주요루틴의 상세 순서도 등을 제시함으로써 쉽게 이해할 수 있도록 하였다.

컴퓨터 도서를 전문으로
항상 연구하며 노력하는 자세로



한영 영한 컴퓨터 용어사전

편집부 편 4×6판 값 4,500원

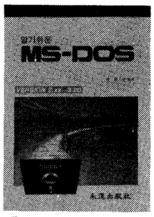
전산, 데이터통신은 물론, 특히 마이크로 컴퓨터 관계의 최신 용어에 역점을 두었으며 권말에 한영색인을 수록하여 사용자 가이드 등 컴퓨터 관계의 원서를 해독하는데 도움이 되도록 엮은 동시에 부록에는 각종 마이컴의 명령 세트 등의 귀중한 연구자료를 수록하여 학생, 실무진 및 연구진을 위한 각종 참고 서겸 용어사전이다.



알기쉬운 Symphony

에디슨 웨슬리 저 김 용성 역 4×6배판 값 7,000원

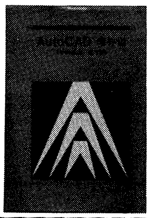
Symphony는 electronic spreadsheet, word processor, database, graph 및 data 통신의 5개 software를 하나의 package로서 실무용 종합 program이다. 즉, Symphony를 사용해서 spreadsheet를 작성하여 data를 입력 조작하고, 문서를 작성 편집, 서식화(format)하고 원격 computer와 교신을 할 수도 있다.



MS-DOS (Version 2.xx～3.20)

이 준구 편저 4×6배판 값 5,000원

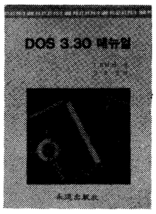
IBM PC XT, AT를 사용하기 위한 기본 도서로 문법에 연연한 설명을 피하고 초보자들도 흥미롭게 MS-DOS를 배울 수 있도록 실습위주로 설명되어 학교나 학원, 회사 등의 교육 교재로 가장 적합하다.



AutoCAD 매뉴얼 (Version 2.18)

Autodesk사 저 국배판 값 12,000원

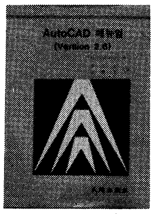
AutoCAD drafting package는 IBM PC 및 그 호환기종에 사용되는 10조 : 1이라는 놀라운 zoom ratio를 갖는 강력한 설계보조용 software로서 우리가 상상할 수 있는 거의 모든 drawing을 신속 정확하게 그려내며, 건축설계, 기기설계, 전자회로설계 등은 물론, freehand sketch에 의한 지도작성, 미술응용, 수학함수의 도표작성 등 광범위한 제도기능을 갖는다.



IBM PC DOS 3.30 매뉴얼

IBM사 저 김 용성 역 4×6배판 값 8,000원

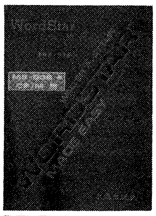
이 책은 본래 DOS 3.30 매뉴얼 원서를 바탕으로 하여 역편된 것이지만, 구 버전 디스켓의 사용자를 위해 버전 3.20 관계 기능도 함께 해설하였으며, 크게 Ⅰ편 DOS User's Guide와 Ⅱ편 DOS Reference로 나뉜다. 특히 프로그래머들이 실무에 있어서 수시로 명령이나 기능에 대한 설명을 쉽게 찾아볼 수 있도록 완전 사전형식으로 엮어진 것이 특징이다.



AutoCAD 매뉴얼 (Version 2.6)

Autodesk사 저 국배판 값 15,000원

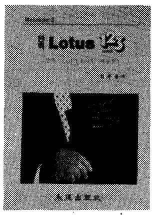
이 책은 IBM PC 및 그 호환기종에 사용되는 10조 : 1이라는 놀라운 Zoom ratio를 갖는 최강의 설계 보조용 S/W인 Auto CAD 버전 2.6을 해설한 기본 매뉴얼로서 구 버전(버전 2.1)에 50여종의 새로운 명령이 추가되었고, 또 기존 명령도 대다수가 크게 보완되었으며, AutoCAD 개정 증보분을 실었다.



WordStar 실무 매뉴얼

Walter A. Ettlin 저 4×6배판 값 4,500원

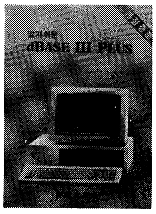
WordStar는 MicroPro International사가 발표한 융통성이 많은 Word Processing Program으로 이 매뉴얼은 일반적으로 사용되는 거의 모든 type의 word processing 기법을 제공해주고 있으며, 특히 이 책에 주어진 연습문제를 완성시킴으로써 얻어지는 기법은 어떤 업무처리용 Word processor에도 활용할 수 있다.



완벽 Lotus 1-2-3 (Release 2)

김 우용 저 4×6배판 값 5,000원

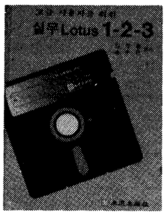
전세계적으로 가장 인기있는 마이크로컴퓨터용 프로그램중의 하나인 Lotus 1-2-3의 가장 최신판인 Release 2를 모델로 초보자도 이해하기 쉽도록 한글과 함께 사용하는 방법을 예를 들어 설명하였다. 1-2-3를 한차원 높게 사용하려는 사람들을 위해서는 내장함수의 사용방법, 매크로 사용법 등을 예를 들어 상세히 설명하고 있다.



알기쉬운 dBASE III PLUS

Robert A. Byers 저 4×6배판 값 6,000원

IBM PC를 중심으로 기술된 Database 입문 과정으로 종래의 dBASE Ⅱ와는 달리 메뉴 방식으로 되어 있어서 명령문 작성의 문법을 암기할 필요가 없이 화면에 나타난 항목 중에서 선택하여 쉽게 사용할 수 있으므로 초보자도 이해할 수 있도록 서술되어 입문과정 교재로 적합하다.



실무 Lotus1-2-3

김 우용 · 홍 수현 편저 260×380m/m 값 10,000원

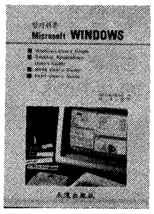
이 책은 사원들의 급여 계산이나 견적서 작성 등 실무에서 활용할 수 있는 프로그램에서부터 경영자 계층에서 경영상태를 분석하고 시뮬레이션할 때의 프로그램에 이르기까지, 지금까지의 방법보다 한차원 높은 단계의 1-2-3 활용 방법을 갈구하던 사용자 계층의 욕구를 만족시켜줄 것이다. 즉시 업무에 활용할 수 있도록 software를 제작 공급하고 있다. (S/W 값 20,000원)



실무 dBASE III PLUS

김 우용 · 김 영미 편저 4×6배판 값 6,000원

이 책은 dBASE Ⅲ PLUS에 대한 기본 지식을 갖춘 사용자가 실무에 쓸 수 있는 프로그램을 직접 작성할 수 있도록 하였으며, 어떤 업무에서나 쓰기 편리한 「기사정보정리」 및 즉시 사용이 가능한 「재무회계」 프로그램을 자세한 설명으로 수록하였다. 특히 Software를 별도로 제작하여 저렴하게 공급하고 있다. (S/W 값 10,000원)



Microsoft WINDOWS

마이크로 소프트사 저 4×6배판 값 7,000원

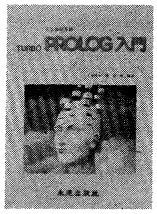
복수의 프로그램을 동시에 사용할 수 있게 해주므로써 마이크로 컴퓨터의 사용자들에게 확장된 DOS라는 평판을 받고있는 WINDOWS 프로그램의 사용방법을 초보자에서부터 한단계씩 단계적으로 학습할 수 있도록 꾸몄다. 한 프로그램에서 다른 프로그램으로 옮겨가는 방법, 데이타를 이송하는 방법 등을 키보드나 마우스를 사용하는 방법과 함께 상세히 설명하고 있다.



dBASE III PLUS 프로그램 모음

이 기성 · 탁 연상 공저 4×6배판 값 9,000원

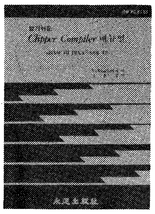
이 책은 한글로 만든 데이타베이스 프로그램(한글 소쿠리), 고객 관리 프로그램, 색인 관리 프로그램, 전화 번호 관리 프로그램, dBASE화일 깨진것 살리기, 문서(저널) 관리 프로그램, 급여 관리 프로그램, 환자(치과) 관리 프로그램, 해운 관리 프로그램 등을 수록하고 별도로 software를 제작하여 공급하므로 업무에 활용할 수 있게 하였다. (S/W 값 20,000원)



인공지능언어 PROLOG 入門

조 동섭 편저 4×6배판 값 5,000원

가장 강력한 인공지능언어로 최근 크게 각광을 받고있는 Prolog 언어의 사용법이 단계적 학습과정을 통해 익힐 수 있도록 꾸며졌다. 컴퓨터 사용자에게 슈퍼컴퓨터와 같은 능력을 갖게 해주는 제5세대 언어인 Turbo PROLOG를 이 책을 학습한 뒤에 스스로 Expert System을 디자인할 수 있는 정도로 숙달될 수 있게 될 것이다.



Clipper Compiler 매뉴얼

Nantucket사 저 김 용성 역 4×6배판 값 5,000원

Clipper compiler로 작성된 program은 dBASE와 같은 interpreter로 작성된 program처럼 각 line마다 error check를 하고 기계어로 변환시킴이 없이 모든 program error를 한꺼번에 식별하고 수정할 수 있어 실행속도가 월등하게 빠른 것이 특징이며, 또한 dBASE Ⅲ PLUS의 command와 함수를 사용할 수 있다.

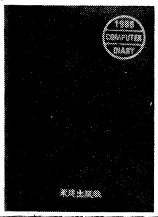


GW-BASIC으로 배우는 인공지능

조 동섭 편저 국판 고급양장 값 4,000원

이 책은 문제해결, 창조성, 초급영어에 의한 컴퓨터와의 대화 등 인공지능을 아는 데 있어서 매우 현실적인 실천에 입각해서 기본이 되는 분야를 소개한 해설서이다.
모든 프로그램은 GW-BASIC에 의해 작성되어 IBM PC/XT/AT 호환기종에서 실행이 가능하도록 하였다.

출판하는 저희「영진출판사」는 컴퓨터 문화창조에 이바지하겠읍니다.



'89 COMPUTER DIARY

편집부 편 4×6배판 고급비닐카바양장 값 3,500원

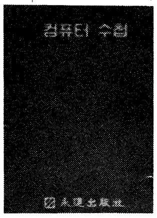
365일 일별메모와 80여 페이지의 메모란, IBM-PC 호환기종을 위한 참고서(MS-DOS, WordStar, Lotus 1-2-3, dBASE Ⅲ Plus, 베이직 명령어 요약표, 한글 DOS 및 한글 카드에서 Graphic Character Cord표), 전국 우편번호, 전국 자동전화번호, 전화번호 메모란, 수도권 지하철 안내도를 수록한 책으로 대량 제작 납품도 가능하다.



XENIX V 매뉴얼

Santa Cruz사 저 김 우용 역 4×6배판 값 7,000원

이 책은 SCO의 XENIX V 2.1.3의 실치에서부터 시스템의 증설 및 변경, 실제 사용시에 부딪치는 문제들에 대한 해결 방법 등, 초기 도입에서부터 실용에 이르기까지 시스템 관리자 및 사용자가 알아두어야 할 사항을 설명하는 기본 매뉴얼이다. 다음으로 Operating System 중 User's Guide Guide와 Reference도 곧 출간될 예정이다.



컴퓨터 수첩 (IBM PC용 APPLE, MSX용)

편집부 편 90×150m/m 비닐카바양장 값 1,200원

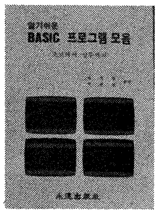
365일 일별메모와 30여 페이지의 메모란, IBM-PC 호환기종을 위한 참고서(MS-DOS, WordStar, Lotus 1-2-3, dBASE Ⅲ PLUS 명령어 및 사용법), 각 기종별 BASIC언어해설, 컴퓨터관련 시험출제기준, 전국 자동전화번호, 전화번호 메모란을 수록한 수첩으로 컴퓨터 학원이나 컴퓨터 메이커용으로 제작 대량 납품도 가능하다.



노턴 유틸리티 4.0 ADVANCED EDITION

탁 연상 편역 4×6배판 값 3,500원

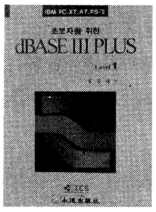
노턴 유틸리티 4.0 ADVANCED EDITION은 지웠던 화일을 다시 복구해 주고, 지웠던 하드 디스크를 다시 복구해 주고, 디렉토리의 화일들의 순서를 마음대로 바꿀 수 있고, 디스크를 검사하고 안전처리해주며, 디렉토리의 구조를 트리구조로 보여주는 등의 다양한 기능을 제공하는 유틸리티 24개로 구성되어 있다.



알기쉬운 BASIC 프로그램 모음

이 기성·탁 연상 공저 4×6배판 값 6,500원

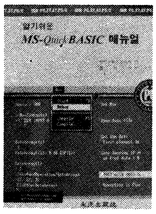
이 책은 개인용 컴퓨터의 사용자들이 가장 쉽게 배울 수 있고, 또 베이직 언어로 유용한 실무용 프로그램을 작성할 수 있도록 각종 실무용 예제를 수록한 베이직 프로그램집이다. 특히 사용자들의 요구에 따라 학습 및 실무에 직접 응용할 수 있도록 software를 제작 보급하고 있다.



초보자를 위한 dBASE III PLUS 1

김 경태 저 4×6배판 값 6,000원

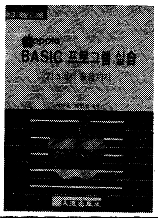
이 책은 dBASE Ⅲ PLUS를 처음 시작하는 분에게 적합하도록 가장 기본적인 사항을 체계적으로 전개하였으며, 과정의 연속성을 유지했으므로 꼭 처음부터 끝까지 읽으면서 dBASE Ⅲ PLUS와 함께 진행하기를 권한다. 곧 Level 2, 3도 출간될 예정이다.



MS-Quick BASIC 매뉴얼

마이크로 소프트사 저 4×6배판 값 7,000원

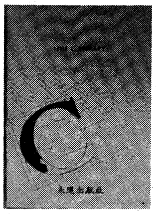
이 책은 마이크로소프트사의 Quick BASIC Version 2.0과 3.0 매뉴얼의 번역서로서, 특히 예제의 이해도를 높이기 위해 한글로 주석을 달았다. Quick BASIC은 기존의 인터프리터 베이직과 거의 완벽한 호환성을 유지하면서도 빠른 속도와 편리한 프로그래밍 환경을 제공하는 강력한 베이직 언어이다.



APPLE BASIC 프로그램 실습

이 기성·탁 연상 공저 4×6배판 값 6,000원

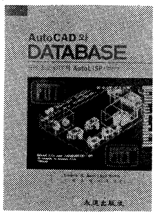
애플 베이직을 사용하여 실제로 프로그램을 작성하는 방법과 실무 프로그램을 보기로 들어서 각 루틴을 자세히 설명하여 왜 이렇게 프로그램을 작성하였는가를 독자가 한 눈에 알 수 있도록 집필하였으며, 현재 실무에서 사용되고 있는 프로그램들이므로 프로그램의 원리를 공부도 하고 실제로 사용할 수도 있어서 일거양득이다.



C 라이브러리

Kris Jamsa저 조 동섭 역 4×6배판 값 5,000원

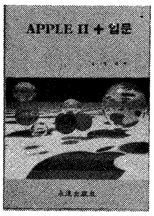
이 책은 C언어로 쓰인 Subroutine이나 함수를 풍부하게 다루고 있다. 이들 Subroutine이나 함수는 모두 범용성이 높기 때문에 어느 분야의 응용 프로그램이나 시스팀 프로그램을 작성하더라도 유용하게 사용될 수 있다. 따라서 C프로그래머에게는 필수적인 참고서가 될 수 있을 것이다.



AutoCAD와 DATABASE

김 용성·서 재철 공역 4×6배판 값 6,000원

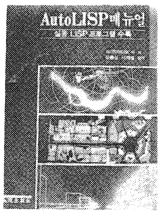
이 책에서는 AutoLISP를 통해서 비 graphic data를 관리하고 조작하는 방법과 DXF file을 조작하여 drawing database를 dBASE나 Lotus 1-2-3와 같은 외부의 database에 link하는 방법을 중심적으로 해설하였다.



APPLE II+입문

송 영태 저 4×6배판 값 3,500원

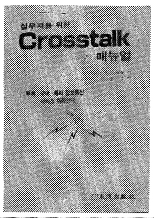
처음 APPLE Ⅱ PLUS를 대하는 초보자가 기본 매뉴얼에서 얻지 못하는 기본 지식에서부터, 주변기기 사용법에 이르기까지 쉽게 설명하였으며, 기본 시스템의 이해와 BASIC 사용에 대한 기본설명과 응용방법, 카세트 레코더와 디스크 드라이브 사용법 APPLE Ⅱ PLUS를 한차원 높여주는 CP/M 사용법에 대하여 명령어의 요약해설과 함께 설명하였다.



AutoLISP 매뉴얼

김 용성·서 재철 편역 국배판 값 7,000원

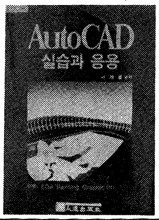
이 책의 제3장에서는 AutoCAD command를 실제로 개발하게 되고, 제4장에서 제6장까지에는 기본이 되는 모든 AutoLISP 함수를 예문을 곁들여 해설하였으며, 제7장에는 AutoLISP 함수를 작성하고 debug할 때 자주 보게 되는 error message를 수록하였다. 부록에는 유용한 많은 실용 program을 수록하여 AutoLISP programming에 있어서 참고가 되도록 하였다.



실무자를 위한 Crosstalk 매뉴얼

Reter W. Gofton저 김용성 역 4×6배판 값 5,500원

이 책은 미국 통신 software의 베스트셀러로서 알려진 Crosstalk XVI의 이용법을 알기쉽게 해설한 것으로, 초심자에서 상급자에 이르기까지 각각의 레벨에서 computer 통신을 실행할 수 있도록 체계적으로 엮어져 있으며, 이 책에 소개하는 command file이나 script file을 이용함으로써 통신비용을 크게 절약할 수 있다.



AutoCAD 실습과 응용

서 재철 편저 국배판 값 10,000원

이 책은 실무에서 AutoCAD를 사용하던 사람이 직접 쓴 책으로 Manual식의 피상적인 접근이 아닌 실전 실습과 응용으로서 높은 부가가치를 창출할 수 있는 AutoCAD의 지침서이다. 특히 EGA PAINT라는 좋은 Graphic Program까지 실어서 AutoCAD의 미비점을 보강하여 Graphic 화면을 최대한 살릴 수 있게 하였다.

신간안내

기초부터 응용사례까지 완전해설
BAR CODE시스템 핸드북
박선우 편저 4×6배판 값 5,000원

해설 **PC 통신 입문 I**
정영일 편저 4×6배판 값 5,000원

프로그래머를 위한
dBASE III PLUS (Level 2)
김경태 저 4×6배판 값 6,000원

전자출판
신문과 출판에서 컴퓨터의 이용
이기성 저 4×6배판 값 8,000원

IBM PC 入門
초보자를 위한 가이드
탁 연상 저 4×6배판 값 4,000원

고급 사용자를 위한
실무 Lotus 1-2-3 (제2집)
홍 수현 편저 8절판 값 12,000원

최첨단 도서 전문 출판
永進出版社
☎ 702-1119(代), 719-1119, 719-1190

무성한 화제속에 드러난 NeXT 컴퓨터시스템

스티브 잡스가 설립한 NeXT사가 마침내 32비트머신「NeXT」를 발표하였다. 1985년에 스스로 창설한 애플사에서 추방된 그가 이번에 발표한 기종은 68030 CPU를 채택하고 오퍼레이팅 시스템에 Mach, 외부기억장치에 리드/라이트 가능한 256M 바이트 광자기 드라이브를 채택한 강력한 기종이다. 대학 등 연구기관에 필요한 교육용 컴퓨터라고 동사가 정의한 NeXT컴퓨터 시스템은 앞으로 10년간의 퍼스널컴퓨터/워크스테이션 시장을 리드하여 갈 것이라고 외국에서는 벌써 기대의 소리가 높아지고 있다. 본지에서는 이러한 NeXT의 전모를 여러 외지를 정리하여 게재한다.

NeXT의 전모

NeXT 컴퓨터는 1피트입방의 메인유니트(소프트웨어를 포함), 17인치 모노크롬모니터, 키보드, 마우스, 레이저프린터(옵션), 330/660M 바이트 HDD(옵션)으로 구성된다. 입방체의 메인유니트는 NeXT의 로고를 의식한 디자인을 채용하고 있다(기본 규격은 표 1을 참조).

메인유니트와 디스플레이의 사이는 3m의 연장케이블로 접속된다. 이 케이블내에 전원라인을 포함하여 모든 신호선이 들어 있어서 다른데에 접속해야 할 케이블은 없다.

메인유니트 후면에는 4개의 NuBus슬롯이 있으며, CPU보드가 이 1개를 점유한다. 나머지 3개 보드의 면적은 일반의 퍼스널컴퓨터용 슬롯보다 크기 때문에 상당한 기능을 탑재한 보드의 설계가 가능하게 된다. 그 외에 전원유니트, 광 자기디스크 드라이브가 콤팩트하게 케이스내에 수용되어 있으며, 옵션의 하드디스크 드라이브를 위한 공간(풀사이즈×2대)도 확보되어 있다.

모니터는「MegaPixel 디스플레이」라고 하며, 사이즈는 17인치이고, 모노크롬 4단계 표시를 실현하고 있다. 해상도는 1120×832도트, 모니터에는 스테레오스피커가 내장되어 있는 외에 오디오용 라인출력단자, 헤드폰단자, 마이크입력단자가 붙어 있다. 컬러표시가 아닌 점을 제외하면 AV(오디오·비쥬얼)지향의 디스플레이라고 하여도 지장이 없을 것이다. 잡스의 설명에 의하면「현재 컬러대응을 검토하고 있다. 그것은 그래픽용 코프로세서탑재의 보드이며 발표는 올해 안으로 될 것이다」고 한다.

심장부에는 25MHz의 68030 채용

일반 사람의 예상대로 CPU에는 모토롤러사의 68030(클럭주파수 25MHz)을 채용하였다. 처리속도는 5MIPS. 애플사의 Mac Ⅱx가 탑재하고 있는 16MHz의 68030과 비교하면 약30% 정도 빠른 것으로 된다. 또 수치연산용 코프로세서는 클럭주파수 25MHz 의 68882를 표준으로 탑재하고 있다.

이 2개의 프로세서 조합으로 현존하는 다른 워크스테이션에 충분히 대항할 수 있는 잠재능력을 기대할 수 있지만 이 머신에는 더욱 고속화를 목적으로 한 칩이 채용되고 있다. 모토롤러사의 56001 DPS가 그것이다. DSP(Digital Signal Processor: 디지탈 시그널 프로세서)는 그 이름과 같이 신호처리를 목적으로 개발된 고속연산칩이다. 일반적으로는 화상처리나 음성의 리얼타임처리 등 특수한 시스템에 사용되고 있는 칩이다. 56001 DSP의 실행속도는 10MIPS이며, 특수한 연산에서는 68030의 명령세트에 의한 연산보다도 50~100배의 속도를 실현한다.

NeXT컴퓨터 시스템에서는 56001 DSP를 배열계산, 팩스/모뎀, 음성신호처리에 사용하고 있다. 교육분야 특히 기술계 분야에서는 이 칩을 사용하고 싶기 때문에 NeXT를 도입한다고 하는 경우도 생길만큼 빠른 실행속도를 자랑한다.

메인메모리는 8M바이트를 표준장비하고 있다. CPU보드상에는 전체로 16M바이트 분의 SIMM 소켓을 마련하고 있기 때문에 8M바이트가 확장가능하다. 표준장비의 8M바이트 중 약 반정도가 유닉스 및 시스템이 점유하고 있기 때문에 실제의 어플리

〈표 1〉 NeXT의 기본규격

CPU	MC68030(클럭 주파수 25MHz)
코프로세서	MC68882(클럭 주파수 25MHz)
DSP	56001DSP(클럭 주파수 25MHz)
RAM	표준 : 8M바이트(최대 16M바이트까지 확장가능)
VRAM	256k바이트
화면표시	17인치 모노크롬(스피커 내장), 해상도 1120×832도트, 4단계 그레이 스케일 표시(디스플레이 포스트 스크립트 대응)
외부기억	표준 : 256M바이트 광자기 디스크×1대(액세스 시간 65ms) 옵션 : 330/660M바이트 HDD를 1대 내장 가능
사운드	샘플링 주파수 44.1kHz, 16비트×2채널 ICP의 3채널(마이크입력×2, 출력×1)이 대응
I/O	NuBus(25MHz)×4, 이더네트×1, 시리얼×2, SCSI×1
프린터	해상도 400bpI, 최대 A4 사이즈
소프트웨어	시스템 : Mach(4.3BSD 콘패치, NFS 서포트) NextStep, Sound & Music Kit, Objective-C, GNU C, Debugger, EMACS, 4.3BSD Utilities Terminal Emulator, DSP Tools. 어플리케이션 : Digital Library, Electronic Mail, Jot, WriteNow, Mathmatica, SQL Database Server Allegro CL Common Lisp.

케이션은 3~4M바이트 정도이다. 또 디스플레이용의 VRAM은 메인메모리와는 다른 공간에 256K바이트를 배치하고 있다.

고속처리를 실현하는 2개의 VLSI

68030, 68882, 56001 DSP라고 하는 3개의 심장부를 지탱하고 있는 것이 신개발의 2개의 커스텀 칩이다. 각각「ICP(Integrated Channel Processor)」와「OSP-(Optical Storage Processor)」라고 하는 이름이 붙어 있다.

앞에서 설명한 바와 같이 동기종은 5MIPS의 68030과 10MIPS의 56001 DSP를 채용하고 있다. 이 MIPS(1초간에 100만회의 연산능력)라는 단위는 시스템의 속도를 아는데에 잘 사용되지만 어디까지나 CPU의 명령실행 속도를 나타내는 것이며 시스템 전체의 속도를 재는 단위는 아니다. 예를 들면 5MIPS의 CPU를 사용한 시스템이어도 CPU의 처리시간의 50%를 I/O처리에 사용하고 있다면 기대할 수 있는 속도는 2.5MPIS로 되어 버린다. 따라서 시스템 속도를 향상시키기 위해서는 CPU가 할 필요가 없는 I/O처리를 할수 있는 만큼 다른데로 분산시켜서 CPU의 부하를 경감시켜야 된다. 대형컴퓨터에서는 이에 대한 처리를 하고 있으며, 거의 모든 I/O기기에 채널 프로세서를 사용하여 기능의 향상을 꾀하고 있다.

NeXT 컴퓨터 시스템의 ICP와 OSP는 대형컴퓨터에서 채용하고 있는 수법을 데스크탑·기종에 응용한 것이다. 동사가 2개의 커스텀칩을「Mainframe on a chip」이라고 부르는 까닭도 여기에 있다.

ICP는 12개의 DMA채널로 구성되어 있다. 12개 각각은 미리 용도가 정해져 있다 (표 2를 참조).

현행의 대부분의 퍼스널컴퓨터에서도 DMA(Direct Memory Aceess)를 디스크드라이브는 화면표시 등의 I/O기기에 사용하는 방법이 취해지고 있지만 이러한 통일적인 수법을 데스크탑 기종에 실현한 것은 동사가 처음이다.

OSP는 ICP의 2채널을 사용하여 스토리지 장치와의 데이터전송을 담당한다. 기본적으로는 1채널은 광 자기디스크용에 사용하며 또 하나의 채널은 SCSI포트용에 사용하고 있다. 또 광 자기디스크의 사용에 불가결인 에러정정기구(ECC)도 DSP에 실현하고 있다.

256M바이트 광 자기디스크 내장

NeXT 컴퓨터 시스템은 대용량을 실현하는 스토리지장치로서 리드/라이트 가능한 광 자기디스크 드라이브 1대를 표준 장비하고 있다. 미디어는 카트리지 방식으로 간단히 착탈할 수 있다(사진 3 을 참조). 카트리지 1매의 기억용량은 256M바이트로 CD-ROM의 약 반이다.

광 자기디스크는 레이저광선에서 미디어상의 자성을 변화시킴으로써 데이터의 리드를 실현하는 것이며, 엄밀하게 말하면 CD-ROM과 같은 광 디스크와는 다르다. 그러나 헤드가 미디어면에 접촉하는 것이 없는 점 등은 같으며, 취급은 매우 간단하고 신뢰성도 높게 되어 있다. 동 드라이브는 캐논제이며, 평균 액세스 시간은 65ms. 동 용량의 하드디스크 드라이브에 비해서 속도가 약 1/4정도 늦지만 앞서 설명한 OSP가 I/O를 담당하기 때문에 CPU에 대한 부하는 거의 없다.

동사는 광 자기디스크 드라이브와는 달리 옵션으로서 330/660M바이트의 하드디스크 드라이브를 지원하고 있다. 양HDD 모두 풀 사이즈이며 본체에 2대까지 내장 가능하다. 인터페이스는 SCSI를 사용한다.

SCSI의 전송레이트는 4.8M바이트/초고속이며 내년중에는 케이스내에 3G바이트의 스토리지 능력을 실현할 예정이다.

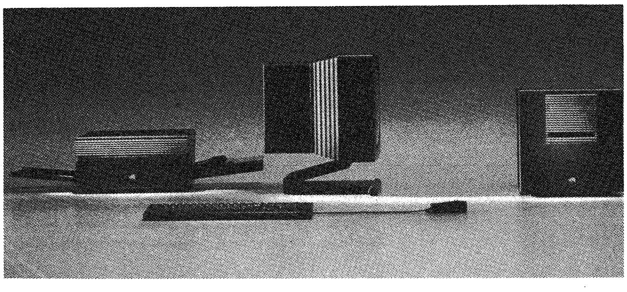
한편 플로피디스크 드라이브는 서포트하고 있지 않다.

충실한 사운드기능

NeXT 컴퓨터시스템은 16비트×2 채널이며, 샘플링주파수 44.1kHz의 스테레오 입력/재생기구를 장비하고 있다. 이 수치는 CD오디오와 같은 수준이다.

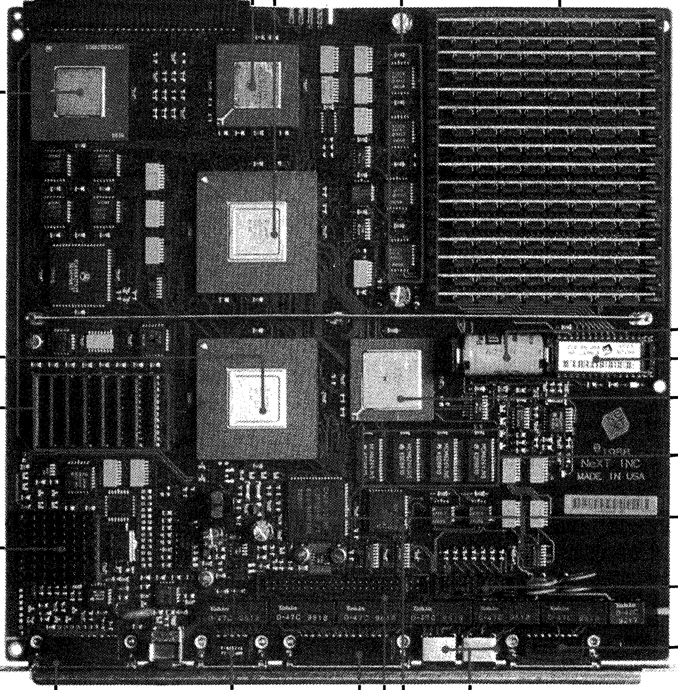
앞에서 설명한 ICP 중의 3채널(2채널이 마이크입력, 1채널이 출력)이 이 때문에 할당되어 있

〈사진 1〉 NeXT 컴퓨터 시스템 규격(메인 유니트(사진우측), 마우스, 17인치 4단계 모노크롬 모니터(사진중앙), 키보드, 해상도 400bpi 레이저프린터(사진좌측)로부터 구성된다.)

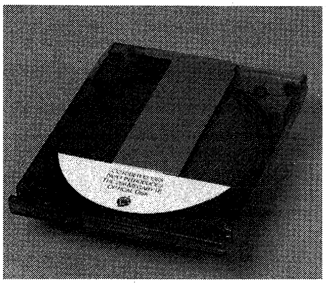


〈표 2〉 ICP의 할당 채널

대 상	DMA 채널 수
이더네트	2
OSP	2
시리얼	2
모니터	1
프린터	1
메모리	1
마이크로폰	2
사운드출력	1



〈사진 2〉 **NeXT의 메인 기판**(중앙에 나열된 거대한 2개의 VLSI(ICP와 OSP)는 후지쯔제. 메인 메모리를 제외하면 주요 칩은 전부 45개 정도이다.)



〈사진 3〉 **용량 256M바이트의 광자기 디스크**(직경은 130㎜. 드라이브의 평균시크 시간은 92ms, 데이터전송속도는 9.1Mbps. 10년 이상이라는 장기간의 데이터 유지자력을 실현하고 있다.)

으며, 더욱 56001 DSP를 사운드 핸들링에 이용할 수 있도록 「SoundKit」와 「MusicKit」라고 하는 소프트웨어 툴을 표준으로 제공한다.

이들을 사용함으로써 DSP를 이용한 고도인 리얼타임샘플링 재생의 응용소프트웨어를 개발하는 것이 가능하게 된다.

그 일예가 소프트웨어 패키지에 포함되는 「NeXT voice mail」이다. 마이크에서 입력한 음성을 디지탈화하여 보이스메일로서

퍼스널컴퓨터의 대부라 불리우는 스티브 잡스

컴퓨터 사용자 가운데서 스티브 잡스의 이름을 알지 못하는 사람은 아무도 없으리라 생각된다. 스티브 잡스와 애플사의 역사는 퍼스널컴퓨터의 역사 바로 그것이라 할 수 있다.

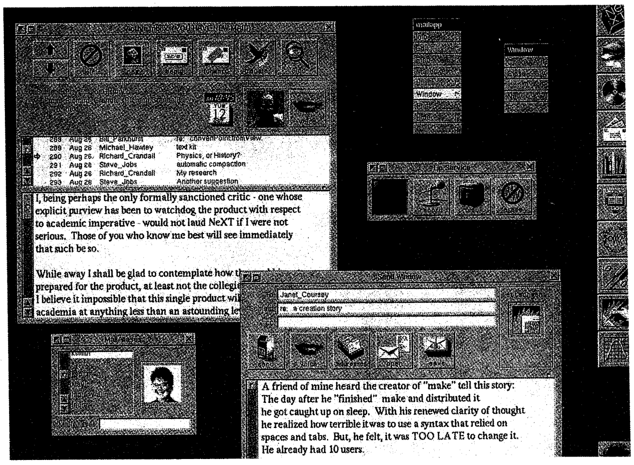
1976년 그는 또 한사람의 스티브인 워즈니악과 함께 애플사를 설립하였다. 그들이 세상에 내보낸 AppleⅡ는 순식간에 세계에 퍼져서 수년의 사이에 퍼스널 컴퓨터업계(생각하면 업계 그것을 창출한 것도 그들 자신이였다)최대의 기업으로 성장하였다. 현재 동사가 판매하고 있는 AppleⅡGS는 CPU가 16비트 판으로 된 것 이외에는 조금도 변화된 것이 없다.

1984년 애플사는 매킨토시를 개발, 발표하였다. 맥은 그 전년에 판매를 개시한 리자의 마이너스 기종이라 할 수 있지만 그래픽을 기본으로 한 유저인터페이스는 당시로서는 충격적인 것이었다. 맥의 개발에 있어서 잡스는 지도적(독재자적이라고도 이야기 된다) 역할을 다하고, 주위의 적지않은 반대를 무릅쓰고 개발을 진행하였다고 한다. 현재에는 맥의 매상은 10억달러, 동사 전체에서는 20억 달러라고 하는 매상을 기록하고 있다.

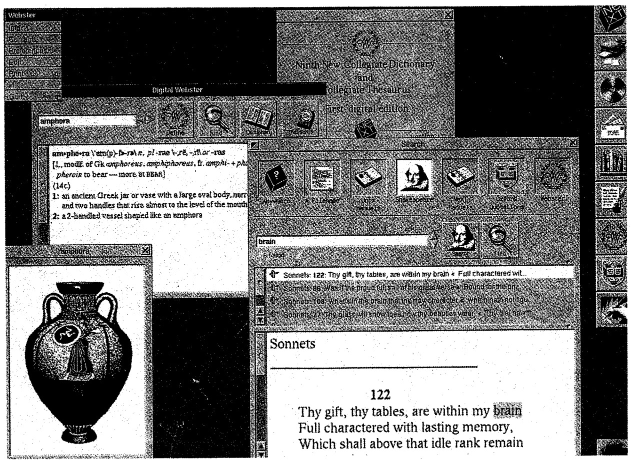
1985년 잡스는 애플사에서 추방된다. 그 후 자기자본으로 NeXT사를 설립, 당초부터 교육시장을 목표로 한 기종개발을 진행하여 왔다.

앞으로 그의 향방이 어떻게 될지는 모르나 현재로서는 NeXT 기종의 발표로 그는 새로운 방향을 모색하고 있는 것임에는 틀림이 없다. 아뭏든 그를 퍼스널컴퓨터계의 대부라고 부르는데 이의를 제기할 사람은 아무도 없을 것이다.

〈사진 4〉 응용 소프트웨어를 동작시킨 화면 (화면은 NeXT Voice mail관련 윈도우를 표시하고 있다.)



〔사진 5〕 윈도우 화면(화면에는 온라인 데이터베이스로 검색 코맨드를 실행한 결과가 표시되어 있다.)

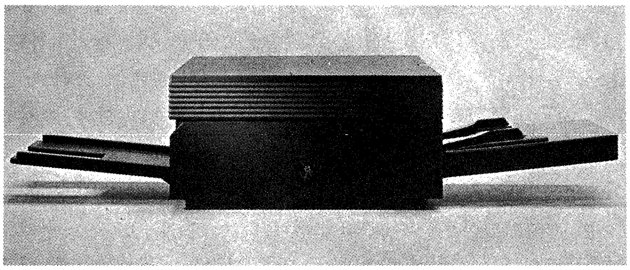


네트워크상에 전송할 수 있다. 수신측은 그 메일을 보이스메일에서 재생하여 듣는 것이 가능하다.

해상도 400dpi의 LBP

레이저프린터는 해상도400dpi로 엔진부분은 캐논제, 케이스사이즈는 매우 콤팩트하며, 물론 포스트스크립트에 대응하고 있다. NeXT 컴퓨터시스템은 디스플레이 포스트스크립트를 표준으로 탑재하고 있기 때문에 디스플레이로 확인한 이미지를 그대로 프린터에 출력 가능하다. 그러나 이 LBP는 현행의 대부분의 포스트스크립트 탑재 프린터와 같이 포스트스크립트 데이터를 처리하는 CPU나 이미지를 전개하는 메모리를 탑재하고 있지 않기 때문에 모든 처리는 메인유니트측에서 한다. 따라서 그만큼 메인유니트의 CPU의 부하가 크게 되는 것이지만 ICP가 충분히 작용하기 때문에 CPU에 대해서는 그렇게 부하가 증대하는 것은 아니다.

〈사진 6〉 해상도 400dpi의 레이저 프린터(엔진부는 캐논제, CPU나 이미지 전개용의 메모리, 포스트 스크립트 드라이버 등은 메인 유니트측에 장비하기 때문에 몸체는 콤팩트하다.)



Mach와 NeXT Step

오퍼레이팅 시스템은 유닉스를 채용하고 있다. 탑재한 유닉스는 카네기메론대학이 개발한 「Mach」.

이것은 UNIX4·3BSD와 호환성을 가지는 평행분산처리이다.

네트워크 환경은 사실상의 업계 표준으로 되어 있는 TPC/IP에 준하며, 선 마이크로시스템사의 NFS(Network File System)를 서포트하고 있다.

NeXT 컴퓨터시스템의 시스템 아키텍처를 그림 1에 나타낸다. Mach의 바로 위에 포스트스크립트 드라이버가 위치한다. 포스트스크립트 드라이버는 화면표시를 제어하는 디스플레이 포스트스크립트의 모두를 처리한다. 포스트스크립트 드라이버의 위에 있는 것이 윈도우서버라고 하는 부분. 2비트/피크셀의 그레이스 케일처리를 포함하는 윈도우 처리는 이 부분에서 제어한다. 이 윈도우시스템은 X-Window 와 호환은 아니다.

그림 1에서 윈도우 서버, 어플리케이션 키트, 인터페이스 필터, 워크스페이스 매니저를 합쳐서 「NextStep」이라고 한다. NextStep 이 동 머신의 중심을 형성하고 있으며, 시스템과 유저의 사이를 연결하는 유일한 맨·머신인터페이스로 되어 있다.

〈그림 1〉 NeXT의 시스템 아키텍처



개발환경으로서는 이하의 것이 시스템에 포함된다.

① Objective-C4.0
② GNU ANSI C 컴파일러
③ GNU C 디버거
④ UNIX 4.3BSD 유틸리티
⑤ 터미널 에뮬레이터
⑥ 윈도우 베이스 텍스트 에디터
⑦ GNU EMACS
⑧ DSP 툴(DSP어셈블러, 디버

거, 배열연산 루틴을 포함한다)

⑨ 사운드키트

⑩ 뮤직키트

동 머신에는 시스템 소프트웨어와 함께 아래에 나타내는 어플리케이션 소프트웨어가 제공된다.

① 보이스메일기능이 붙은 UNIX mail「Electronic Mail」

② 텍스트베이스의 간이 데이터베이스이며, 메모장으로서도 사용할 수 있는「Jot」

③ 매킨토시용으로 판매되고 있는 영문 워드프로세서를 이식한「WriteNow」(판권은 잡스가 가지고 있다)

④ 그래픽베이스의 수치계산시스템「Mathmatica」

⑤ Sybase사의 데이터베이스「SQL Database Server」

⑥ Franz사의 Common Lisp「Allegro CL common Lisp」

IBM사와의 협력체제

동사가 IBM사와 공동으로 무엇인가의 작업을 하고 있다고 하는 소문은 신기종의 발표 이전부터 흘러 나오고 있었다.

잡스가 동 기종의 중심을 형성하고 있는 NextStep을 IBM사에 라이센스공여한 이유는「계약은 많은 소프트웨어 개발자를 지원하는 것이며 NeXT용으로 개발한 소프트웨어는 동시에 NextStep 을 탑재한 IBM의 머신유저도 대상으로 할 수 있기 때문이다」라고 한다.

IBM사에서는 동사의 유닉스인 AIX를 Interactive Systems 사와 공동개발중이다. 대상으로 되는 기종에는 PS / 2, IBM RT, IBM 370 등을 들 수 있다. 그리고 AIX가 완성되면 이들의 기종에 NextStep이 탑재될 예정으로 있다고 한다.

잡스가「NeXT 컴퓨터시스템은 어디까지나 고등인력을 위한 교육시장용 머신」이라고 강조하는 이면에는 IBM사가 비지니스 분야에서 NextStep의 시장개척을 한다고 하는 암묵의 양해가 있는 것 같다.

저가격으로 판매시작

시스템의 가격은 256M바이트의 광 자기드라이브를 내장한 메인유니트와 MegaPixel 디스플레이, 시스템 소프트웨어나 어플리케이션 소프트웨어를 제공하여 6,500달러, 레이저빔 프린터가 2,000달러로 경이적인 가격을 실현하고 있다(표 3을 참조). 교육용의 바겐세일 가격이라고 생각하여도 될 정도이다.

〈표 3〉 시스템의 가격

메인 유니트	8M바이트 RAM 256M바이트 광자기 드라이브×1대	6500달러
17인치 CRT	해상도 1120×832도트, 4 단계	
소프트웨어	Mach와 개발툴, 어플리케이션	
프 린 터	해상도 400bpi 레이저 프린터	2000달러

동사의 발표에 앞서서 애플사는 Mac Ⅱx를 발표하였다. Mac Ⅱx도 68030을 사용하고 있지만 클럭주파수가 16MHz판이라는 점 때문에 기대된 만큼의 속도는 바랄 수 없을 것 같다. 다만 A / UX 대응이라는 것과 같은 시장을 대상으로 하고 있다는 점 때문에 NeXT의 강력한 경합기종으로 되는 것은 불가피하다.

애플사에서 추방된 스티브 잡스가 오랜 침묵의 끝에 제시한 회답이 NeXT 컴퓨터 시스템이다. 전체적으로는 신기술을 개발하였다고 하기 보다도 현재의 최고 기술을 콤팩트하게 더구나 누구나가 납득하는 가격으로 완성하였다고 보는 쪽이 옳다. 물론 사운드 기능의 충실이나 하이퍼 카드식의 새로운 아이디어를 풍성하게 도입하고 있는 점은 주목할 가치가 있다. 발표후의 업계의 반응은「지금 즉시라도 구입하고 싶다」고 하는 긍정적인 의견이 압도적으로 많다.

한편 컬러그래픽 기능의 결여 등을 지적하며「기대는 벗어난 기종」이란 결론을 내는 관계자도 있다. 컬러 기능을 추가하면 적어도 1만달러 상당의 코스트업으로 되는 것은 틀림없다. 덧붙여서 말하면 출하시의 0.8버전을 올 2 / 4분기 안으로 정규의 1.0버전으로 번전업하여 시판할 예정이라 한다.

동 기종이 잡스가 기대하는 바와 같이 1990년대의 퍼스널컴퓨터 / 워크스테이션의 주도적 지위를 구축하기 위해서는 더욱 큰 관문을 뚫고 나가지 않으면 안될 것이다. PC

NeXT사가 겨냥하는 교육시장

어떤 조사보고에 의하면 미국의 고등교육기관의 퍼스널컴퓨터의 사용목적은 90%이상이 워드프로세서용, 60%가 데이터베이스 화일관리, 그리고 50%가 스프레드시트 처리로 되어 있다. 단지 워드프로세서로서 사용하는 것이면 퍼스널컴퓨터에 조금도 구애되지 않는 것이지만 최근 유행의 데스크탑·퍼블리싱을 하게 되면 워크스테이션 상당의 기종에 레이저프린터라고 하는 조합이 필요하게 된다.

현재 미국의 대학에서 가동하고 있는 워크스테이션은 거의 100%가 UNIX를 채용하고 있다. OS / 2는 아직도 보급에 시간이 걸릴 것 같고, IBM사가 AIX노선을 본격화시키게 되면 장래적으로는 불안이 남는다. 도대체 IBM사가 PS / 2용으로 왜 XENIX를 채용하지 않았을가는 지금도 수수께끼이다. 또한 UNIX는 Sun Microsystems 사, Apollo사, DEC사 등 많은 머신에서 가동되고 있는 점도 무시할 수 없을 것이다. 또 미국의 대학 컴퓨터 사이언스 학과에서는 반드시라고 하여도 좋을 정도로 UNIX를 정규수업에 추가하고 있어 매년 UNIX의 신망자가 속속 배출되고 있는 것이다. 고등교육의 현장에서는 UNIX머신에의 수요가 더욱 더 확대되는 경향에 있다.

미국에는 3000을 넘는 대학이 존재하고, 관계자는 학생도 포함하여 1,000만인 이상에 달한다고 한다. 이들의 기관 혹은 개인이 구입하는 컴퓨터의 전체액은 1,000억 달러를 넘는다. 여기에 NeXT사가 목표로 하고 있는 거대한 시장이 잠재하고 있다.

구입가이드

비디오 보드

비디오 보드는 화면에 문자나 그림이 나타나게 해주는 것으로 텍스트모드와 그래픽모드 2가지의 모드방식으로 문자나 그림을 처리하고 있다.

텍스트모드는 텍스트 버퍼에서 2바이트를 읽어 그 문자에 대한 이미지를 만들어 내어 모니터에 나타내고, 그래픽모드는 그래픽 버퍼에서 1바이트를 읽어 그 바이트내의 각 비트가 "1"이면 "점"을, "0"이면 아무 것도 나타내지 않는다(칼라일 경우는 그 점의 칼라를 나타내는 정보를 읽어서 처리한다).

문자와 그림은 비디오 램의 영역(도스에서 사용자 메모리 외의 특별한 목적으로 사용되는 영역)을 비디오 보드내의 디스플레이 콘트롤러(모토롤라에서 제작한 6845 CRTC라는 칩)가 읽어들여 그 내용을 나타낸다.

비디오 보드는 크게 모노와 칼라 그래픽보드 두부분으로 나뉘는데 모노에는 MDA, HGC가 칼라에는 CGA, EGA, VGA 등이 표준방식으로 사용되고 있다.

MDA는 텍스트모드로 표시문자수가 80✱25이며, 1문자당 셀이 9✱14의 도트를 가진다. 이 보드는 그래픽처리가 되지 않는다.

HGC는 텍스트모드에서는 MDA와 같고 그래픽모드에서는 720✱348이라는 고해상도를 지원하고 있다.

CGA는 텍스트모드에서는 40✱25행과 80✱25행이 둘다 가능하며 또한 1문자당 8✱8의 도트를 가지며 16가지의 색상을 사용할 수 있는데 한문자 셀내에 배경색으로 8가지색을, 경계색으로 16가지 색상중 1가지를 선택하여 사용할 수 있다.

중해상도의 그래픽모드에서는 320✱200의 해상도와 한 화면상에서 4가지 색상을 사용할 수 있다. 또한 16가지 색상중 1가지를 배경 색상으로 선택하고 3가지 색상을 지정하여 사용할 수 있으며 1문자당 셀이 8✱8도트를 차지하고 팔레트중 3차 색상(흰색이나 브라운색)에서만 40✱25행의 텍스트를 사용할 수 있다.

고해상도의 그래픽모드에서는 640✱200의 해상도로 내용은 흰색으로 배경색은 검은색으로 표시되고 80✱25행의 텍스트모드가 사용 가능하고 1문자당 셀은 8✱8이다.

EGA는 EGA, CGA, MDA에서 돌아가는 소프트웨어를 사용할 수 있을때 EGA와 CGA, MDA 모드를 사용할 수 있도록 되어 있어서 보드상의 딥 스위치를 바꾸면 CGA로만 짜여진 소프트웨어에서도 작동이 가능하다.

그리고 딥 스위치를 바꾸는 불편함을 없애기 위해서 이를 소프트웨어적으로 구현해 자동적으로 체크해 주는 자동스위치 방식의 EGA호환보드들도 있다. EGA는 램의 비디오 버퍼를 64K, 128K, 256K 등의 메모리를 선택하여 사용할 수 있어 EGA 본래의 기능인 비디오 버퍼와 모노크롬과 칼라 비디오용 버퍼를 모두 이용할 수 있도록 되어 있다.

MDA 모드에서는 텍스트모드가 같고 그래픽모드에서는 640✱350의 해상도로 색의 밀도나 깜박임 등의 9가지 기능을 이용하고 조합하여 화면상에서 4가지의 의사색상을 사용하도록 되어 있다. 문자는 80✱25행과 1문자당 셀이 8✱14 도트를 가진다.

CGA 모드에서는 텍스트, 중해상도, 고해상도에서의 모드가 CGA와 같고 한 화면에서 16가지 색상을 중해상도와 고해상도에서 사용할 수 있도록 한 점이 강화되었다.

EGA 모드의 텍스트모드에서 한문자 셀은 8✱14의 도트수를 가지고 경계색은 없다. 중해상도와 고해상도에서는 16가지 색상을 한 화면에 사용할 수 있으며 640✱350의 해상도를 지원한다. 또 64KB 비디오램에서는 16가지 색상중 4가지 색상을 사용할 수 있으며 128KB나 256KB에서는 64가지 색상중 16가지 색상을 사용하고 80✱25행의 텍스트를 사용하며 1문자당 셀이 8✱14의 도트를 차지한다.

VGA도 EGA와 마찬가지로 비디오 표준방식인 MDA, HGC, CGA를 지원하고 EGA모드까지 지원한다. VGA는 텍스트모드에서 132✱44행을 사용할 수 있고 266, 144의 팔레트와 256가지의 배경색을 지원한다. 해상도는 800✱600과 1280✱600, 640✱480, 800✱560, 1024✱768, 800✱560 등 각 사마다 다양한 종류가 있다.

EGA나 VGA는 모노크롬과 호환성이 있으므로 본래의 그래픽모드에서 직접적으로 비디오 램을 사용할 수 없고 비

디오 시스템의 특별한 로직을 이용한다.

VGA는 초기에 BIOS의 호환성만으로 설계되어서 IBM의 표준을 대체할 수 없었다. 그러나 뒤이어 나온 비디오 보드들은 레지스터의 호환성을 가지게 되어 IBM PC의 어느 기종에서도 실현할 수 있게 되었다. 이 레지스터의 호환으로 BIOS의 루틴을 거치지 않고도 훌륭한 색상, 더 많은 글자를 화면에 나타낼 수 있었던 것이다.

또 하나 부가적인 비디오 보드는 PGA 보드로 이것은 16만가지의 팔레트 중에 256가지의 색상을 한 화면에 표시할 수 있으며 640×480의 해상도를 제공한다.

여기서 비디오 보드를 사용할 때 주의할 점은 모노크롬을 사용하던 사용자가 칼라로 대치하고자 할 때 컴퓨터 본체의 마더보드상에서 칼라를 지정해야 한다. 특히 IBM PC 호환기종들은 만든 회사마다 칼라를 각각 다르게 지정해 주어야 하며 이러한 칼라를 지정하려면 딥 스위치를 세팅하거나 점퍼를 이용하기도 한다. 자세한 사항은 사용하고 있는 매뉴얼을 참조하거나 구매한 회사에 의뢰하면 된다.

한글카드

국내에 IBM PC 16비트 컴퓨터가 소개될 즈음에 그것과 호환성을 가지는 제품들은 한글이 가능한 소프트웨어가 보급되지 못해서 마땅한 기반을 잡지 못했다.

그러나 7비트의 완성형 한글카드가 개발되어 로터스 1-2-3과 데이터베이스 시리즈등이 한글용 소프트웨어로 대체되면서 일반업무의 전산화가 용이하게 되자 차츰 복제품들은 활기를 찾는다.

대부분의 영문 소프트웨어들은 BIOS 콜을 이용하지않고 텍스트 버퍼에 정보를 바로 저장하는 방법을 이용하고 있으며 이 방법은 속도가 빠르며 프로그래머의 입장에서 보면 유용한 점이 많아 널리 사용되고 있으며 바로 이러한 방법이 "한글카드"의 탄생을 가져오게 된다.

한글처리 방법

컴퓨터에서 한글을 처리하고 있는 방법으로는 소프트웨어적인 방법과 하드웨어적인 방법이 있다.

소프트웨어적인 방법은 텍스트 화면상태에서 하는 것이 아니라 그래픽 화면에 한글의 모양을 출력하기 때문에 텍스트 모드보다는 속도가 느리다.

그리고 한 바이트로는 한글을 표현할 수 없어서 2바이트의 코드로 한글을 표시한다. 2바이트는 한글/영문 표시부분과 초성, 중성, 종성의 네부분으로 구성되는데 초성, 중성, 종성의 모양을 모아 합성시켜 한글을 만든다. 실제로는 여러개의 폰트들이 있어서('가'에서의 초성 ㄱ과 '각'에서의 초성 ㄱ은 글자모양이 다르다) 보다 예쁜 한글을 만든다.

하드웨어적인 방법은 별도의 카드를 꽂는 방법으로 소프트웨어적인 방법과 같이 PC의 CPU가 직접 작업하지 않으므로 처리속도가 빠르고 텍스트 모드에서도 한글을 구현할 수 있는 잇점이 있다. 그러나 이것은 2바이트의 코드를 롬에 저장하여 확정된 한글의 모양을 가리키게하여 한글카드가 그것을 텍스트 화면에 출력시켜 주기 때문에 처리속도가 증대되는 대신에 한글의 모양을 롬에 저장시키므로 구현할 수 있는 글자수가 한정되어 있다는 단점을 가지고 있다.

한글카드의 종류

한글카드에는 크게 조합형과 완성형의 2가지가 있으며 조합형은 한글을 만들어주는 작업을 하드웨어(한글카드)가 해주기 때문에 텍스트화면에서 그 처리 속도가 빠르다.

조합형은 모든 한글을 표시할 수 있다는 장점이 있지만 영문 소프트웨어를 수정해서 써야한다는 단점이 있다.

완성형은 7비트와 8비트가 있는데 7비트는 각 바이트의 최상위 바이트를 사용하지 않으며 한글의 모양을 미리 완성된 형태로 보관 하였다가 사용한다. 이 7비트 완성형은 국내에서 된 것이 아니고 2바이트의 특수 코드를 처리하기 위해 미국에서 제작한 것으로 최상위 비트를 사용하지 않기 때문에 한글의 처음과 끝을 나타낼 수 없어서 모든 문자를 표시할 수 없다. 7비트의 완성형카드는 텍스트 화면상에서 실행되는 모든 소프트웨어상에서 수정없이 한글을 사용할 수 있도록 되어 있다.

이것의 장점은 대개의 업무용 소프트웨어에서 소프트웨어를 고치지 않고도 한글카드만 꽂으면 바로 한글을 사용할 수 있다는 것이다. 단점으로는 표현할 수 있는 한글의 수가 한정되어 있다는 것이다. 즉 영문 소프트웨어에서 별나른 문제없이 사용하려면 1400자 이하의 한글코드만 사용해야 하는데 한글을 사용하려면 적어도 2300자는 되어야 한글을 무리없이 사용할 수 있기 때문이다.

7비트에 비하여 8비트 완성형은 한글이 2350자, 한자 4888자를 지원하면서 사용자 정의용 한글과 한자도 가능하도록 되어 있다. 이 형태의 한글코드는 국가기간 전산망의 한글표준형코드로 채택되어 사용되고 있다.

현재 시판되고 있는 한글카드는 초기에 각기 다른 기종들이 서로 다른 한글과 한자를 사용하도록 되어 있어서 사용자로 하여금 혼란을 주었으나 요즘에는 대체로 완성형과 조합형을 선택하여 사용할 수 있도록 지원하고 있어 사용자에게 편리함을 제공하고 있으며 대체로 그래픽모드에서도 텍스트처리가 가능하도록 되어 있다.

멀티 펑션카드

보드 하나로 다양한 기능을 제공하는 카드를 말하는 것으

로 내부 기억장소(128K~640K바이트 램), RS-232C비동기 직렬통신 인터페이스, 센트로닉스와 호환성이 있는 병렬프린터 인터페이스나 플로피디스크 인터페이스, 게임 인터페이스 등 다양한 기능을 지니고 있다.

멀티펑션 카드는 지니고 있는 기능에 따라 가격에서 차이가 나는데 유저가 원하는 기능에 적합한 카드를 구입할 경우에는 각각의 기능을 가진 카드를 사용하는 것보다도 훨씬 경제적일 수 있다. 세운상가에서 현재 판매하고 있는 멀티펑션카드의 경우 서비스 문제나 기술지원 사항등을 고려하여 신중하게 선택하여야 한다.

멀티펑션 카드에서 기능을 선택할 때에는 딥스위치나 점퍼를 이용하여 조정할 수 있으며 단, 주의해야 할 사항으로 점퍼나 딥스위치의 조작시에는 항상 전원은 "OFF"상태를 유지해야 한다는 점이다.

통신카드

컴퓨터의 활발한 보급과 함께 그의 활용범위 또한 다양해지고 있다.

근래에는 원거리에서도 시간과 거리에 구애받지 않고 원하는 정보를 얻기 위한 컴퓨터간의 통신이 대두되고 있다. 개인용컴퓨터끼리의 정보교환이라든가 대형컴퓨터의 정보서비스 프로그램과의 연결로 유저는 원하는 항목의 정보를 얻을 수가 있게 되었다.

개인용컴퓨터를 보유하고 있는 유저들이 통신을 시작하고자 할때에는 통신장비(모뎀, 케이블, 코넥터) 등이 필요하며 전송매체로 공중전화선을 이용하여 통신을 시작할 수 있다.

통신장비를 선택하기에 앞서 표준 인터페이스로 되어 있는 RS-232C 규격에 대해 알아보고, 실제 PC에서 어떻게 이용되고 있는지에 대해서 살펴보자.

통상 우리들이 일컫는 RS-232C는 미국의 EIA(Electronic Industries Association)에서 제정한 규격을 말하는 것으로 직렬 이진데이터의 교환을 하는 데이터 터미널장비(DTE : 터미널이나 PC)와 데이터통신장비(DCE : 모뎀)간의 인터페이스의 제반사항을 규정한 것이라 할 수 있다.

즉, RS-232C는 DTE, DCE와 함께 데이터 통신망을 효율적으로 구성하는 중요한 요소라 할 수 있다.

한편 RS-232C인터페이스를 이용하여 데이터를 전송하기 위해서는 통신하고자 하는 두시스템 간에 다음과 같은 항목이 일치되어야 한다.

●통신형태의 지정

●반이중방식(Half Duplex) : 데이터의 전송에 있어 양쪽 방향으로 신호의 전송이 가능하기도 하나 어느 순간에는 반드시 한쪽 방향으로 전송이 이루어진다.

●전이중방식(Full Duplex) : 동시에 양쪽 방향으로 신호의 전송이 가능하며 만약 두개의 채널이 준비되어 있다면 한 채널로 송신하는 동안 다른 채널로 수신할 수 있다.

●동기방식과 비동기방식

●동기방식 : 대형컴퓨터와 퍼스컴간의 통신에 많이 쓰인다.

●비동기방식 : 조보동기방식 이라고도 하며 송신절차가 간단하여 퍼스컴간의 통신방식을 말한다.

●패리티 검사

데이터의 전송시에 에러를 검사하는 수법으로 수신측에서 미리 약속된 패리티 방식대로 비트가 수신되었는지를 매 바이트마다 검사한다.

●시작비트와 정지비트

한문자의 선두에 한비트로 된 스타트 비트와 최후에 1비트로 된 패리티 비트와 두비트로 된 스톱비트를 부가해서 송신한다.

●보오율(Baud Rate)

통신속도를 말하며 접속하는 양쪽 시스템의 성능과 통신선로의 특성에 크게 좌우된다.

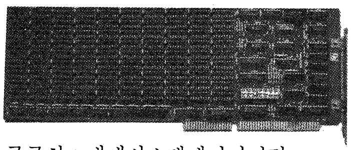
1초간에 송수신 가능한 비트수를 말한다. 일반적으로 직렬형 비동기식 통식 방식에는 300, 600, 1200, 2400, 4800, 9600 보오의 속도를 사용한다.

●데이터 전송단위

한편 PC 구입시 직렬통신 카드가 내장되어 있거나 멀티펑션카드의 내장으로 통신포트를 지원할 경우에는 상관이 없지만 기능을 보유하고 있지 않는 경우에는 통신카드를 구입한다. 현재 세운상가에서 판매하고 있는 통신카드의 종류에는 여러가지가 시판되고 있고 1포트일 경우 2~3만원선, 2포트의 경우에는 3~4만원, 4포트인 경우에는 10~15만원선에서 구입하여 사용할 수 있다. 만일 2대의 PC끼리 20~30m정도의 단거리에서 모뎀을 이용하지 않고 통신을 하려고 하는 경우에는 두개의 직렬 인터페이스 포트를 RS-232C케이블로 직접 연결할 수 있는데 이러한 케이블을 널 모뎀(Null Modem)케이블이라 하며 유저가 직접 제작하여 사용할 수 있지만 세운상가에서 판매하고 있는 것을 구입하여 간단히 접속하여 사용할 수도 있다. 가격은 1만원정도이며 이 경우에는 비동기 통신용 패키지가 필요하고 MS-DOS용으로 PC TALK Ⅲ, QMODEM, TRANSCEND, PC COMPLETE, SMARTCOM Ⅱ, MITE, SYMPHONY, dBASE Ⅲ+, CROSSTALK Ⅷ 등이 있다.

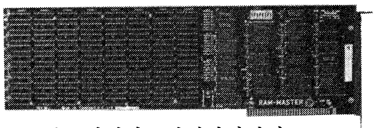
메모리 카드

DSRC



- 공급처 : 대세시스템엔지니어링
  (02) 679-1161
- 특　징 : 최고 2.5MB까지 확장할 수 있으며, IBM PC / AT 및 386기종과 호환기종에 쓰인다.

EMS-2000



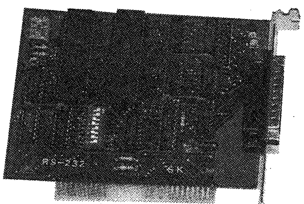
- 공급처 : 대세시스템엔지니어링
  (02) 679-1161
- 특　징 : 최고 2MB까지 확장가능하고 IBM PC / XT / AT 및 386기종과 호환기종을 대상으로 하고 있다.

TG-286Ⅲ MC 카드

- 공급처 : 삼보컴퓨터
  (02) 3902-114
- 특　징 : CPU보드 자체에 640KB시스템 메모리용량을 사용자가 원하는대로 내장하고 추가로 확장할 필요가 있을때는 IMB 또는 2MB의 TG-286Ⅲ MC카드를 선택할 수 있다.

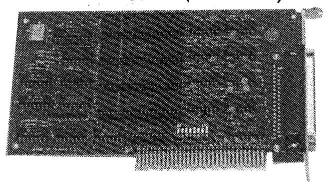
통신카드

RS-232C 카드(2포트)



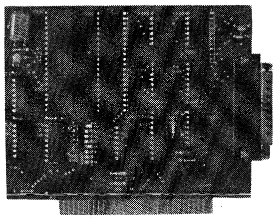
- 공급처 : 원컴퓨터
  (02) 272-6486
- 특　징 : IBM 호환기종간의 직렬 통신이 되며, 대형 컴퓨터의 터미널로도 사용할 수 있다. 모뎀과 연결하여 원거리 통신도 가능하다. 마우스, 플로터 등과 별도의 인터페이스 카드 없이 연결할 수 있으며, 2포트로 확장이 용이하다. 전송속도는 110 보오에서 9600보오까지 자유로이 변환할 수 있다. (IBM-PC / XT / AT용)

RS-232C 카드(4포트)



- 공급처 : 원컴퓨터
  (02) 272-6486
- 특　징 : ZENIX, MS-DOS 및 OS용으로 사용하며, 4대의 터미널로 연결하여 원거리통신도 가능하다(IBM-PC / AT용).

RS-232카드



- 공급처 : 대세시스템 엔지니어링
  (02) 679-1161
- 특　징 : 2직렬 포트

RS-232C카드(2포트)

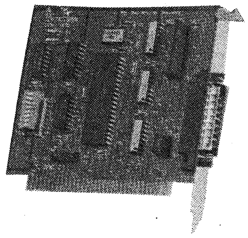
- 공급처 : 한진양행
  (02) 274-2540
- 특　징 : 직렬 비동기방식, COM1, COM2

RS-232C카드(4포트)

- 공급처 : 한진양행
  (02) 274-2540
- 특　징 : 직렬비동기방식, COM1, 2, 3, 4

RS-422카드

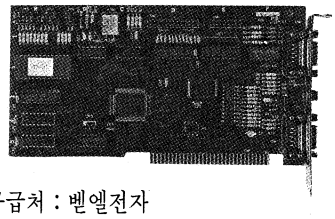
- 공급처 : 원컴퓨터



  (02) 272-6486
- 특　징 : EIA RS-422 표준, 비동기통신 프로토콜, 문자길이 : 5, 6, 7 or 8, 스톱비트 : 1, 1.5 or 2, 패리티 : odd, even or no, 통신속도 : 50~56,000보오

멀티펑션카드

G7-C

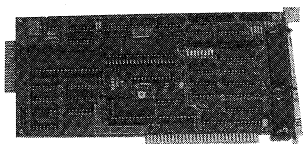


- 공급처 : 벤엘전자
  (02) 275-2887
- 특　징 : IBM 컬러 디스플레이 어댑터 호환(부가기능 : 320×200모드에서 4가지 색상), 허큘리스 그래픽 어댑터 호환, 16가지 색상의 640×400 고해상도 컬러모니터 연결, 프린터포트*, 직렬 RS-232C 포트*, 리얼타임 클럭, 플로피디스크 드라이브 콘트롤러*, 라이트 펜 인터페이스, 게임 포트, 384K바이트 확장RAM

(*은 옵션)

멀티펑션카드

- 공급처 : 현대전자
  (02) 741-1311
- 특　징 : RS-232C 통신포트, 리얼타임 클럭, 게임포트



멀티펑션카드

- 공급처 : 원컴퓨터

(02) 272-6486
- 특 징 : 2채널 RS-232C 인터페이스, 게임인터페이스, 리얼타임클럭 / 캘린더, 플로피디스크 인터페이스(IBM-PC / XT용)

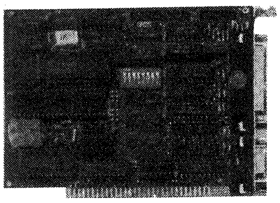
멀티 펑션카드

- 공급처 : 슈퍼컴퓨터
  (02) 277-2042
- 특 성 : 모노크롬, 조이스틱, 직렬 RS--232C, FDC포트, HDC포트, 리얼 타임 클럭, 패러럴 포트

MSC—5603

- 공급처 : 삼성전자
  (02) 559-2626
- 특 징 : 3포트
  (직렬 3개, 터미널 3개)

멀티 I/O Plus 카드



- 공급처 : 원컴퓨터
  (02) 272-6486
- 특 징:2개 RS-232C 직렬포트, 게임 포트, 병렬 프린터 포트

비디오카드

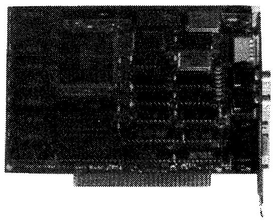
MVA 카드

- 공급처 : 현대전자
  (02) 741-1311
- 특 징 : CGA, MDA, HGC와 호환, 옵션 및 프로그램에 따라 스위치로 조정, 다양한 모드 제공, 단색 텍스트 모드(80×25), 단색 그래픽 모드(720×348), 칼라 텍스트모드(80×25), 132열모드(로터스 1-2-3, 사이드 킥, 수퍼 키를 사용), 칼라그래픽모드(16색상 640×200, 320-×200)

EGA카드

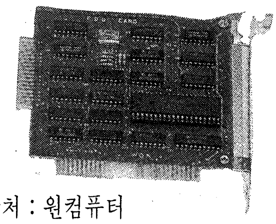
- 공급원 : 금성사
  (02) 493-6611
- 특 징 : 고해상도를 필요로 하는 업무에 적용, 16가지 색상사용, 해상도 (640×350, 640×480, 800×600)

EGA 카드



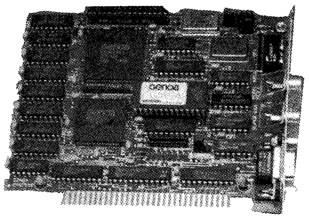
- 공급처 : 대세시스템 엔지니어링
  (02) 679-1161
- 특 징 : IBM PC EGA와 호환, 840×480의 해상도

EGA 카드



- 공급처 : 원컴퓨터
  (02) 272-6486
- 특 징 : 칼라그래픽으로 사용, 고해상도(640×350, 640×480), IBM-PC / XT / AT용

Super EGA HiRes +™

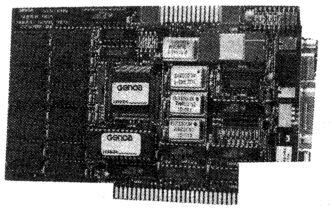


- 공급처 : (주) 태일 시스템
  (02) 756-5340
- 특 징 : IBM EGA와 호환, 최대 해상도 800×600, 오토캐드, 윈도우, 젬, 벤츄라, 로터스 등의 드라이버들을 포함, 텍스트 모드 : 영문 표준모드

Orchid ProDesigner VGA Plus™

- 공급처 : 삼부 엔터프라이즈사
  (02) 678-7457
- 특 징 : VGA, MCGA, EGA, CGA, MDA, HGC, 레지스터와 BIOS의 호환, 기본 메모리 512K

제노아 슈퍼 VGA



- 공급처 : (주) 태일시스템
  (02) 756-5340
- 특 징 : IBM, VGA, EGA와 호환, 1024×768 해상도, 최대 256칼라, 아날로그와 TTL 모니터를 모두 지원, 오토캐드, 윈도우, 젬, 벤츄라, 로터스 등의 S/W 툴 제공

한글카드

한글 CGA 카드

- 공급처 : (주) 금성사
  (02) 493-6612
- 특 징 : 모노와 칼라 그래픽 지원, 원보드의 종합적인 기능 600×400의 고해상도, 16가지 색상지원, 텍스트모드(영문 80×25, 한글 40×25)

MDA—KS

- 공급처 : (주) 옴니테크
  (02) 546-4266
- 특 징 : 행정 전산망용 완성형코드(옴니 EGA 호환), IBM PC용 MDA와 호환, 사용자 정의문자 사용 가능, 행정전산망용 완성형 한글한자 확장 BIOS제공, 640×400과 텍스트 모드제공

EGA—KS

- 공급처 : (주) 옴니테크
  (02) 546-4266

• 특 징 : PC-XT / AT / 386에서 사용할 수 있음, 영문 EGA와 호환성을 가짐, CGA, EGA, VGA 모드를 제공, 행정 전산망용 한글과 한자의 확장 BIOS의 규격을 지원함, 그래픽모드에서도 텍스트 처리가 가능함, KSC 5601 완성형 코드 채택, 사용자 정의문자를 사용할 수 있음 (188자), 그래픽모드(640✱480✱16색상, 640✱350✱16색상(EGA), 640✱200 모노크롬(CGA), 320✱200✱4색상(CGA)), 텍스트모드(영문 : 80✱25, 132✱25, 40-✱25, 한글 : 40✱25, 66✱25, 20✱25)

CGA—KC

• 공급처 : (주) 옴니테크
(02) 546-4266
• 특 징 : IBM PC용 CGA와 호환, 640×400의 고해상도 그래픽(흰색, 검은색), 320×200, 640×200 그래픽 모드 오토매틱 버티칼 더블스캐닝 기능, 비디오 초기화 스크램블 롬을 사용하므로 각종 모니터에 인터페이스 가능, 로터스, 디베이스, 워드스타의 영문 패키지를 수정없이 그대로 사용(7비트 2바이트 한글), 5170자의 완성형 코드의 한자사용.

모노 한글 Enhancement 카드

• 공급처 : (주) 삼보컴퓨터
(02) 3902-114
• 특 징 : 기존의 모노 그래픽보드에 접속하여 사용(피지 백 카드(Piggy Back Card) 교체시), 조합형과 완성형을 스위치나 소프트웨어적으로 선택하여 사용, 유저 폰트 라이트용 16KB SRAM(512 캐릭터 용량)이 있어 유저가 정의하여 폰트를 사용할 수 있음, 그래픽 패키지에서 한글을 구현할 수 있음, KS 표준한자 4888자와 KS 그래픽코드(특수부호)를 제공, 구점입력기능, 옵션 파라미터 변경기능

현대 한글 한자카드

• 공급처 : 현대전자
(02) 741-1311
• 특 징 : 한국표준규격의 행정전산망용 한글사용, 2바이트 완성형, 한글 / 한자 처리 BIOS를 롬에 내장하여 주기억장치를 적게 사용하면서 빠른 처리 속도유지, 한글 / 한자 및 특수문자 내장, 문자 폰트를 사용자가 정의, 2바이트 완성형 한자 사용, 2바이트 조합형, 해상도(640×400, 720×348), 한글 MS-DOS 및 영문 MS-DOS 동시 사용

한글 행정 전산용 카드

• 공급처 : 한진양행
(02) 274-2540
• 특 징 : HGA 한글, 행정 전산망 전용 (한글, 한자, 그래픽 처리가 가능), 마우스, 라이트 펜, 프린터 포트의 기능 포함.

책을 마감하고

●황금주사위는 던져지고
선택이라는 기로에서
조건이란 과연 무엇일까?
어김없이 89년은 다가오고
365일이라는 날들에 나의 의지를 담아보자.
〈鉉〉

●무엇인가에 실패했다는 생각이 들때 호된 따귀 한대를 기꺼이 용납하려는 마음을 가질 수 있어야 하겠지요. 그러나 마음만큼 행동으로 말할 수 있는 상태는 좀처럼 만나지지 않습니다. 조금 더 덮어줄 수 있는 용기가 모자란 탓이겠지요.

89년이라는 호된 따귀 한대가 이번에도 찾아왔읍니다. 자, 괜찮습니다. 산다는 일이 원래 그런 것. 〈仙〉

●둘!
고향이 없는 사람은 없겠지?
나도
고향에 있는 나의 보금자리를 사랑한다.
언제나 포근한 나의 보금자리를…
"Happy new Year!" 〈榮〉

●「日日是好日」인 己巳年이길…
해마다 이맘때면 새로운 마음가짐으로 시작했고, 모든 시간이 흐른 뒤에 돌아다 본 나의 모습은 또 그저 그런 나날을 보낸 후회로 자꾸만 초라해진다. 하지만 새해의 시간을 보낸 뒤에는 자신의 초라함을 보기 보다는 자신의 성장을 기쁨으로 느끼기 위하여 항상 새롭게 태어나려는 노력을 게을리 하지 말자고 다짐해본다.

독자여러분들도 매일 매일 그저 그런 나날이 아닌 「日日是好日」(날마다 좋은날, 새로운 날)인 한해가 되십시오. 〈七〉

●가려합니다.
가고있읍니다.
갔읍니다.
그는 갔지만 새로운 이를 위해 포근한 마음으로 빈자리를 채우렵니다. 새해에도 P.C 친구들 복 많이 받으세요!〈賢〉